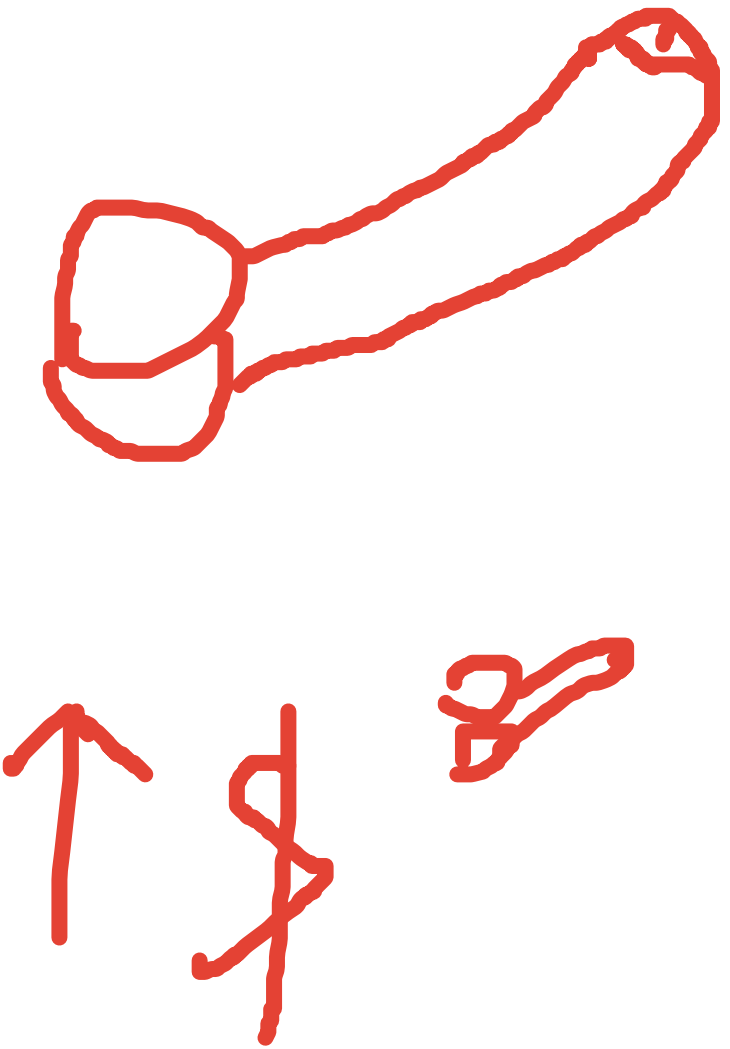
SEBITA MALDONADO Y
CARLA VAIRETTI

# ANALYTICS Y BIG DATA

## CIENCIA DE LOS DATOS APLICADA AL MUNDO DE LOS NEGOCIOS

RiL editores



# Analytics y Big Data

Sebastián Maldonado y Carla Vairetti

# Analytics y Big Data

## Ciencia de los Datos aplicada al mundo de los negocios

"No basta con que estudien, deben también dejar de ser idiotas ignorantes"

658.40 Maldonado, Sebastián
M Analytics y Big Data. Ciencia de los Datos aplicada al mundo de los negocios / Sebastián Maldonado y Carla Vairetti. – – Santiago : RIL editores, 2022.

404 p. ; 23 cm.
ISBN: 978-84-18982-63-7

1 BIG DATA. 2 PROCESAMIENTO ELECTRÓNICO DE DATOS. 3. TECNOLOGÍA DE LA INFORMACIÓN.

Analytics y Big Data.
Ciencia de los Datos aplicada al mundo de los negocios
Primera edición: marzo de 2022

© Sebastián Maldonado y Carla Vairetti, 2022
Registro de Propiedad Intelectual
N° 2021-A-7943

© RIL® editores, 2022

Sede Santiago:
Los Leones 2258
CP 7511055 Providencia
Santiago de Chile
(56) 22 22 38 100
ril@rileditores.com • www.rileditores.com

Sede Valparaíso:
Cochrane 639, of. 92
CP 2361801 Valparaíso
(56) 32 274 6203
valparaiso@rileditores.com

Sede España:
europa@rileditores.com • Barcelona

Composición e impresión: RIL® editores
Diseño de portada: Matías González Pereira

Impreso en Chile • *Printed in Chile*

ISBN 978-84-18982-63-7

Derechos reservados.

# Índice

Prefacio ..... 11
Referencias ..... 15
Agradecimientos ..... 17

1. Introducción a la analítica de negocios ..... 19
1.1 Capacidades analíticas ..... 20
1.2 Dimensión técnica: ciencia de los datos ..... 23
1.3 Dimensión práctica: vínculo con aplicaciones y analítica de negocios ..... 27
1.4 Proceso de extracción de conocimiento ..... 29
1.5 Dimensión práctica: lenguaje de programación Python ... 36
Referencias ..... 37

2. *Descriptive Analytics* y preprocesamiento ..... 39
2.1 Análisis exploratorio de datos (AED) ..... 41
2.2 Limpieza de datos ..... 54
2.3 Transformación de datos ..... 60
2.4 Selección de atributos ..... 66
Referencias ..... 73

Laboratorio 1: Preprocesamiento ..... 75
1. Subir un archivo de Google Drive ..... 75
2. Análisis exploratorio de datos (AED) ..... 76
3. Limpieza de datos ..... 83
4. Transformación ..... 85
5. Filtros (selección de atributos) ..... 88
6. Descargar base de datos preprocesada ..... 93

Aplicación 1: Segmentación de clientes y *Targeting* ..... 95
1. Segmentación utilizando análisis RFM ..... 95
2. Segmentación utilizando análisis RFM ..... 100
Referencias ..... 105

3. *Diagnostic Analytics* ..... 107
3.1 Inteligencia de negocios, visualización y *Dashboarding* 108
Metodología ..... 134
Referencias ..... 134

Laboratorio 2: Segmentación de clientes y *clustering* ... 137
1. K-medias 1: regla de codo ..... 138
2. K-medias 2: solución con cuatro *clusters* ..... 139
3. K-medias 3: solución con cinco *clusters* ..... 142
4. Fundamentos de las bases de datos ..... 145
4.1 Bases de datos y su diseño ..... 145
Referencias ..... 183

5. *Big Data Analytics* ..... 185
5.1 Las 8 V´s del *Big Data* ..... 185
5.2 Hadoop ..... 187
5.3 SQL en Hadoop y *apps* ..... 189
5.4 Apache Spark ..... 192
5.5 Herramientas de Big Data para la empresa ..... 195
Referencias ..... 197

Aplicación 2: Asignación de créditos ..... 199
1. Definición de variable objetivo y predictores ..... 201
2. Inferencia de los rechazados ..... 201
3. *Weight of Evidence* ..... 202
4. Construcción de una *Scorecard* ..... 203
Referencias ..... 205

6. *Predictive Analytics* I: modelos transparentes ..... 207
6.1 Introducción ..... 207
6.2 Regresión logística ..... 208
6.3 Regresión lineal y su adaptación a clasificación ..... 209
6.4 Interpretación y selección de atributos ..... 212
6.5 Árboles de decisión ..... 216
6.6 Validación de modelos ..... 225
6.7 Medidas de desempeño ..... 228
6.8 Desbalance de clases ..... 234
Referencias ..... 236

Aplicación 3: Fuga de clientes ..... 237
1. CRM analítico ..... 237
2. LTV y beneficio de una campaña de retención ..... 240
Referencias ..... 244

7. *Predictive Analytics* II:
Modelos «caja negra» tradicionales ..... 245
7.1 K vecinos más cercanos ..... 246
7.2 Naive Bayes ..... 249
7.3 Random Forest ..... 255
7.4 Interpretando modelos de caja negra I: BORUTA ..... 259
7.5 Extensión multiclase ..... 264
Referencias ..... 268

Aplicación 4: Experiencia del consumidor (CX) ..... 271
1. Fundamentos de la experiencia
y satisfacción del consumidor ..... 272
2. Inteligencia artificial aplicada a CX ..... 274
3. Propuesta para el análisis de datos de CX con IA ..... 275
Referencias ..... 284

8. *Predictive Analytics* III: Inteligencia artificial ..... 287
8.1 Definición de una red neuronal ..... 287
8.2 Algoritmo backpropagation ..... 291

8.3 Actualidad en los modelos de deep learning ..... 308
Referencias ..... 314

Laboratorio 3: Clasificación ..... 315
1. Separación de datos ..... 316
2. Entrenamiento y validación ..... 317
3. Evaluación ..... 328

9. *Predictive Analytics* IV: Regresión y series de tiempo ... 337
9.1 Introducción ..... 337
9.2 Modelos de regresión ..... 338
9.3 Medidas de desempeño para salida continua ..... 342
9.4 Modelos de serie de tiempo ..... 344
Referencias ..... 357

10. *Prescriptive Analytics* ..... 359
10.1 Optimización y simulación ..... 359
10.2 Decisiones de umbral y medidas de *profit* ..... 367
10.3 Aprendizaje causal ..... 373
Referencias ..... 382

Laboratorio 4: Predicción y decisión de umbral ..... 383
1. Preprocesamiento de datos de predicción ..... 384
2. Predicción ..... 386
3. Exportar datos a Excel ..... 388

11. Conclusión ..... 389
11.1 Implementación y monitoreo de modelos ..... 390
11.2 Mitos y leyendas de la inteligencia artificial ..... 396
11.3 Recursos adicionales ..... 401
Referencias ..... 403

# Prefacio

A veces me pregunto si alguien aquí va a recordar algo de lo que digo. No les tengo mucho cariño, de hecho los desprecio, pero quisiera caerles bien. - Sebastián Maldonado Alarcón

Las empresas enfrentan importantes desafíos para monetizar su información. El fenómeno mundial del Big Data, caracterizado por un aumento sin precedentes en la cantidad de información disponible, que estima alcanzar unos 175 zettabytes en 2025 (Reinsel, Gantz & Rydning, 2018), viene acompañado con un mayor nivel de complejidad debido a las diferentes fuentes de información, tales como sensores de dispositivos móviles, imágenes, texto, etcétera.

El mundo de los negocios no ha estado ajeno a esta tendencia, siendo las empresas las principales beneficiadas gracias a las nuevas oportunidades que ofrece la analítica de negocios *(business analytics)* en la toma de decisiones. Una muestra de la relevancia que tiene la analítica de negocios es el estudio de la consultora McKinsey que plantea que «aquellas organizaciones que adoptan analítica de negocios como cultura tienen 23 veces más probabilidades de adquirir clientes, seis veces más probabilidades de retener a esos clientes y 19 veces más probabilidades de ser rentables» (Bokman *et al.*, 2018).

Este libro tiene por foco enseñar técnicas analíticas mediante ejemplos aplicados a negocios. La teoría se presenta en capítulos destacados en **azul**, que buscan entregar al profesional de negocios una caja de herramientas lo suficientemente amplia como para cubrir el espectro de desafíos de la analítica de negocios. Este libro se diferencia de otros sobre Big Data empresarial que solo buscan contar historias interesantes y atractivas para motivar, convenciendo sobre la necesidad de estudiarla. A la vez, se diferencia de aquellos libros que presentan métodos y teoría de forma genérica y con un elevado tecnicismo. Nuestro libro se encuentra en un lugar intermedio, pues está escrito de forma concisa y solo incluye aquellas fórmulas que son imprescindibles para la comprensión de las técnicas.

Su objetivo es acercar al lector a las aplicaciones de negocio, por ello se presentan diferentes tareas de la analítica en secciones dedicadas. Estas se encuentran intercaladas con la teoría, con el fin de explicar mejor aquellos métodos que dan una mejor solución a los problemas de negocios. Estas secciones se encuentran destacadas en **rojo**.

Este es un libro *hands-on*, es decir, nos ensuciaremos las manos con códigos en el lenguaje de programación Python. A modo de laboratorio, aplicaremos nuestra caja de herramientas para dar solución a desafíos de negocios, extrayendo conocimiento relevante para la toma de decisiones. Esta es la mejor forma de aprender las implicancias del modelo e interpretar los patrones encontrados. Estos códigos son explicados en las secciones dedicadas en **verde**, y se incluyen además en la página web del libro.

El libro se encuentra dirigido a analistas o consultores que deseen profundizar en técnicas avanzadas de analítica de negocios. A su vez, puede resultar muy interesante para estadísticos e ingenieros con conocimientos en técnicas analíticas que deseen aprender cómo estas son útiles en el mundo de los negocios. Si bien puede sonar avanzado en su diseño, no requiere conocimientos previos más que nociones básicas de análisis de datos (estadística simple y manipulación de datos en planillas) y manejo básico de herramientas computacionales.

El capítulo 1 introduce este libro, motivando las aplicaciones de negocio y explicando los diferentes conceptos que han formado parte de este mundo en las últimas décadas. Adicionalmente, presenta las diferentes capacidades analíticas descritas por la consultora Gartner (Sapp *et al.*, 2019), las cuales guían los diferentes niveles de madurez y adopción de técnicas analíticas en la empresa. Estos niveles también representan, hasta cierto punto, el orden de los capítulos de este libro.

El capítulo 2 discute la primera y más simple capacidad analítica: *descriptive analytics*. El objetivo de la analítica descriptiva es dar respuesta a la pregunta «¿qué pasó?», entregando un análisis retrospectivo de los datos. Además de discutir estadística descriptiva, este capítulo presenta las técnicas más importantes para preprocesar

los datos y corregir errores en ellos. Antes de este capítulo se presenta la primera aplicación de negocio (perfilamiento de clientes) y concluye con un análisis descriptivo y el preprocesamiento de datos de clientes como primer taller.

El capítulo 3 está dedicado a la segunda capacidad analítica: *diagnostic analytics*, que busca responder la pregunta «¿por qué pasó?». Esta capacidad analítica busca identificar relaciones entre variables y establecer causalidades. En este capítulo se discuten técnicas avanzadas de interpretación de datos y selección de atributos, las cuales complementan el preprocesamiento. Adicionalmente, se presentan técnicas de segmentación o *clustering* y reglas de asociación. El capítulo concluye con el taller que soluciona la problemática de perfilamiento de clientes en la industria bancaria.

Antes de continuar con las capacidades analíticas, profundizaremos en las técnicas de administración y almacenamientos de datos, las cuales son clave para la analítica de negocios. El capítulo 4 presenta las técnicas clásicas de almacenamiento de datos (las bases de datos relacionales), las cuales siguen plenamente vigentes en las empresas. El capítulo 5 está dedicado a la evolución de estas estrategias: las herramientas computacionales de Big Data que son capaces de manejar grandes volúmenes de información compleja y de diferentes fuentes.

Los cuatro capítulos siguientes presentan la tercera capacidad analítica *(predictive analytics)* desde diferentes ángulos. La modelación predictiva se introduce en el capítulo 6 con métodos interpretables clásicos como la regresión logística y los árboles de decisión, y asignación de crédito *(credit scoring)* como aplicación. Esta última se encuentra fuertemente regulada, por eso requiere el uso de modelos interpretables.

El capítulo 7 muestra el uso de modelos de «caja negra» o no interpretables como alternativa a tareas de clasificación. Modelos populares de aprendizaje de máquinas, tales como los *k* vecinos más cercanos o *random forest,* se presentan en este capítulo. Como aplicación elegimos la predicción de fuga de clientes, la cual es una tarea de marketing fundamental dentro de las empresas ya que permite

diseñar políticas de retención de clientes efectivas. Adicionalmente, esta aplicación cuenta con la libertad de poder aplicar cualquier modelo predictivo, a diferencia de la tarea de asignación de créditos.

El capítulo 8 presenta los recientes avances en métodos de inteligencia artificial para predecir tareas de negocio con datos más sofisticados que los tradicionales utilizados para describir a los clientes. En particular, se discuten las redes neuronales artificiales y el aprendizaje profundo como avances fundamentales en el modelamiento de datos como texto e imágenes. Este capítulo presenta como aplicación la experiencia del consumidor (*Customer eXperience*, o CX), un área del marketing dedicada a estudiar las percepciones de un cliente después de interactuar con una empresa. Datos en formato de texto —tales como encuestas, comentarios, reclamos y cualquier otro tipo de retroalimentación— representan una fuente valiosa para entender a los clientes, y su análisis automático representa uno de los mayores niveles de adopción y madurez en una empresa en relación con las capacidades analíticas. El capítulo finaliza con el taller de clasificación predictiva, en donde se comparan diferentes técnicas transparentes, de caja negra e inteligencia artificial (IA), eligiendo, finalmente, la que tiene un mejor desempeño predictivo.

El último capítulo, dedicado a *predictive analytics,* es el correspondiente a regresión y series de tiempo. A diferencia de la tarea de clasificación discutida en los capítulos 6 al 8, el capítulo 9 está dedicado a tareas predictivas numéricas (en vez de categóricas).

El capítulo 10 presenta la cuarta y última capacidad analítica, conocida como *prescriptive analytics*. La analítica prescriptiva combina las tres capacidades anteriores junto con herramientas de optimización y modelamiento matemático para lograr la automatización de decisiones. En este capítulo se discutirán varios casos de éxito que presentan esta fusión de áreas con el fin de que las empresas funcionen mejor, alineando la analítica con los objetivos de negocio. Este capítulo finaliza con un último taller relacionado con la aplicación del mejor modelo predictivo en nuevos datos, y cómo podemos mejorar la toma de decisiones con estrategias simples de modelación.

Finalmente, el capítulo 11 representa el cierre de este libro, donde se discuten diferentes aspectos relevantes de la analítica de datos e inteligencia computacional. Dentro de estos se consideran la implementación y monitoreo de modelos, cuestiones éticas y aplicaciones fuera del mundo de los negocios.

## Referencias

Bokman, A., Fiedler, L., Perrey, J. & Pickersgill, A. (2018). *Five facts: How customer analytics boosts corporate performance*. McKinsey & Company. Recuperado de: https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/five-facts-how-customer-analytics-boosts-corporate-performance

Reinsel, D., Gantz, J. & Rydning, J. (2018). *The Digitization of the World from Edge to Core*. Recuperado de: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

# Agradecimientos

Antes que todo, me gustaría agradecer a mi mismo, por todo el esfuerzo que me tomó llegar hasta acá. Este libro no hubieseido posible sin el apoyo de nuestras afi-liaciones: la Facultad de Ingeniería y Ciencias Aplicadas de la Universidad de los Andes (Carla Vairetti) y el Departamento de Control de Gestión y Sistemas de Información de la Facultad de Economía y Negocios de la Universidad de Chile (Sebastián Maldonado). Agradecemos también el valioso aporte del centro de invstigación al cual pertenecemos: el Instituto Sistemas Complejos de Ingeniería (PIA/ BASAL AFB180003).

No me agradecen. Nadie dice "hasta luego, profe", nadie dice "gracias". Se van como si yo fuera un semáforo. Funcional mientras dura, invisible cuando ya pasó. Está bien. No vine aquí a hacer amigos. Vine a... bueno. No sé exactamente a qué vine. Pero estoy aquí. Y sé más de Business Intelligence que cualquiera en esta sala, incluido yo hace veinte años.

Este libro está dedicado a todos mis estudiantes a lo largo de veinte años de docencia. No porque me hayan inspirado —en general, no lo hicieron— sino porque su persistente mediocridad me hizo entender que la claridad pedagógica no es una virtud: es una necesidad de supervivencia.

estos años que hemos compartido e intercambiado ideas, que nos han llevado a ser los académicos que hoy somos. Agradecemos especialmente a Francisco Herrera-Triguero y todo su grupo de la Universidad de Granada (España), con quien hemos explorado varios temas en *Analytics* y Big Data en los últimos años.

(Sebastián Maldonado) Quiero reconocer a mi «Doktorvater» Richard Weber, por haberme llevado al camino de la analítica de negocios y por todos estos años de consejo y amistad.

(Carla Vairetti) Quiero agradecer especialmente a quien me ha apoyado y guiado con sus sabios consejos desde mis comienzos como académica, Orazio Descalzi.

Finalmente, nos gustaría agradecerte a ti, que has adquirido este libro. Hemos hecho un gran esfuerzo por hacerlo ameno, deseando que puedas aprender mucho de él. Por eso, ¡tu opinión nos importa! Cualquier comentario es bienvenido. Siéntete libre de ponerte en contacto con nosotros a través del sitio web del libro:

www.vairetti-maldonado.cl

Sebastián Maldonado y Carla Vairetti

# 1. Introducción a la analítica de negocios

Vivimos en un mundo cuya digitalización y creación de inmensos volúmenes de datos aumentan casi en tiempo real. Diversos artículos y estudios de parte de organizaciones como IBM, quienes mencionan que más de 150 zettabytes (150 billones de gigas) requerirán de distintos análisis para 2025, o Forbes, quienes descubrieron que se creaban 2,5 trillones de bytes de datos cada día (Amor, 2020), lo demuestran.

Lo anterior no solo corresponde a un fenómeno moderno que la sociedad en su conjunto está viviendo, constituye también un desafío crítico que las organizaciones deben enfrentar para generar soluciones basadas en datos y, con ellas, ventajas competitivas que mejoren su posición en la industria respectiva.

Con la misma velocidad, se ha potenciado la utilización de diversas metodologías y disciplinas para llegar a estas soluciones y ventajas competitivas. Del mismo modo, las herramientas y *softwares* que sirven como motor y piedra angular de cada una de ellas han surgido, y/o se han modificado, para estos fines.

Un claro ejemplo de esto es el lenguaje de programación Python —que será utilizado en las secciones prácticas del libro—, que, dadas sus características y ventajas específicas, ha adaptado y generado diversas herramientas para el desarrollo de análisis y soluciones basadas en datos.

Podemos afirmar que las metodologías y las herramientas computacionales que dan soporte a las prácticas analíticas han evolucionado dramáticamente, al mismo tiempo que se ha producido un aumento en el volumen de datos. En este punto es importante

preguntarnos, ¿quiénes llevarán a cabo estas prácticas?, ¿cómo serán explotadas las metodologías y las herramientas novedosas?

Al pensar en las respuestas a estas preguntas, resulta evidente que haya un aumento de la demanda de profesionales y técnicos capacitados para poder trabajar, gestionar y obtener los mejores resultados posibles a partir de datos. La falta de profesionales y la escasez de talento especializado han sido dos de los principales obstáculos que ha enfrentado la industria.

Así, el propósito de este libro es disminuir las brechas generadas, tanto en profesionales actuales como futuros, en lo relativo a sus capacidades analíticas, gestión de datos y en su potencial desempeño en procesos de ciencia de datos. Lo anterior se realizará desde una mirada técnica y práctica, dando a entender de manera sencilla los modelos, algoritmos, técnicas y herramientas que forman parte de estos procesos, con el objetivo de potenciar las habilidades de quienes realizan proyectos de ciencia de datos de manera temporal y de aquellos que pretenden dedicarse de manera permanente a esta disciplina.

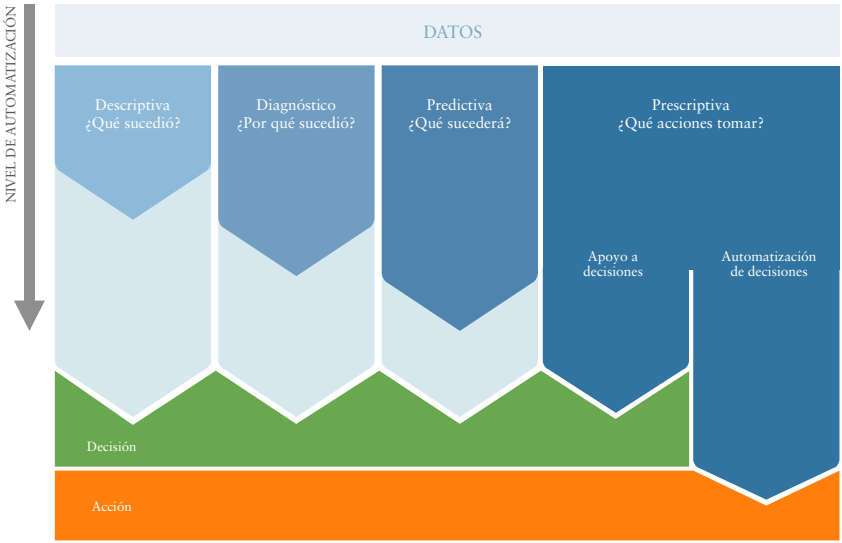
## 1.1 Capacidades analíticas

Otro de los puntos que se pretende abarcar a lo largo del libro, corresponde al cómo los profesionales deben utilizar los datos con distintos objetivos, entre los cuales destacan adquirir, organizar, analizar y entregar información valiosa para la toma de decisiones o creación de ventajas competitivas.

En este sentido, el componente «analizar» puede percibirse como una parte simple, sin embargo, con la evolución de la disponibilidad de información y de los volúmenes de datos existentes, esta se ha vuelto mucho más demandante y requiere de ciertas capacidades analíticas para lograr el objetivo planteado. Según el artículo «2019 Planning Guide for Data and Analytics» (Sapp *et al.*, 2019), la analítica de negocios considera cuadro capacidades primarias: (1) analítica descriptiva, (2) analítica de diagnóstico, (3) analítica predictiva y (4) analítica prescriptiva. Cada una hace referencia a distintas

preguntas y a sus respectivas respuestas, y, además, requieren de distintas decisiones y acciones a realizar. Estas cuatro capacidades se presentan en la figura 1.1.

Figura 1.1: Capacidades analíticas




En primer lugar, se encuentran las capacidades descriptivas y de diagnóstico, las que sirven de ayuda para responder las preguntas con una mirada retrospectiva de los datos. La capacidad descriptiva representa la base de cualquier tipo de análisis de datos y, por ende, ningún proceso ni herramienta puede considerarse como íntegra sin ella. Permite acceder a información básica o específica, dependiendo del contexto y del problema de negocio a analizar, para así poder responder a preguntas relativas al análisis mismo, tales como ¿qué sucedió?, ¿cómo sucedió? o ¿cuándo sucedió? El análisis descriptivo de datos permite identificar, además, problemas con los datos disponibles, tales como la presencia de inconsistencias o valores perdidos.

La capacidad conocida como *diagnostic analytics*, por su parte, permite responder **por qué** sucedió algo en específico. Lo anterior requiere de análisis más sofisticados que los descriptivos, involucrando la interacción de varias variables o factores que pueden influir en un determinado fenómeno. Modelos econométricos o de regresión

son ejemplos de enfoques multivariantes que buscan responder ciertas preguntas en base a los datos. Otro ejemplo corresponde a modelos de segmentación de clientes, que buscan identificar perfiles accionables, es decir, grupos de clientes con características comunes a los cuales podría tratar de forma especial como empresa (clientes *premium*, con riesgo de fuga, etcétera).

Por otro lado, se encuentran las capacidades predictiva y prescriptiva, las que apuntan a visualizar los posibles resultados o escenarios futuros del análisis. La capacidad de análisis predictivo permite identificar patrones o tendencias en los datos disponibles que puedan ser extrapolables a futuro. Asumiendo que el pasado es un buen predictor del futuro, podemos inferir comportamientos de clientes u otros (ventas futuras, participaciones de mercado, etcétera), con el fin de anticiparnos a necesidades. Un ejemplo clásico es el *credit scoring* o asignación de créditos, que consiste en predecir si un cliente será un «buen pagador» y cumplirá sus obligaciones financieras en caso de recibir un crédito o si, por el contrario, será un «mal pagador» y caerá en *default*. Las políticas de otorgamiento de crédito de cualquier banco o institución financiera se rigen por este tipo de modelos. Ejemplos como este se verán en detalle a lo largo del libro.

Por último, la capacidad prescriptiva es la que responde la pregunta más desafiante: ¿qué debería hacer? Para responder esta pregunta, se deben combinar estrategias predictivas (ya sea basadas en inteligencia artificial o Big Data) con técnicas que permitan la automatización de decisiones (Hitner, 2017). Estas últimas suelen venir del mundo de la investigación de operaciones y buscan modelar matemáticamente las decisiones, con el fin de cuantificar sus desenlaces y elegir la estrategia óptima. En el libro se discutirán estrategias de optimización y simulación para la toma de decisiones.

Dado el actual escenario de competencia y búsqueda constante de ventajas competitivas, las organizaciones deben proveer cada vez más apoyo para el desarrollo de las capacidades analíticas, ya sea con herramientas de tecnologías de información (TI), recursos financieros, capacitaciones, etcétera.

El objetivo del libro es apoyar la preparación y la mejora de estas capacidades analíticas, a través de cada una de las técnicas y aplicaciones que serán descritas, de manera tal que los profesionales del área sepan cómo integrarlas a sus procesos analíticos. De esta manera es posible generar el impacto esperado en organizaciones tanto desde una dimensión técnica, como de la dimensión práctica de la ciencia de datos.

## 1.2 Dimensión técnica: ciencia de los datos

Para entender de mejor manera algunos de los conceptos mencionados al principio del capítulo, tales como ciencia de datos, Big Data o aprendizaje automático, entre otros, es necesario comprender que existe un «mundo de las técnicas» que engloba diferentes estrategias para el análisis de datos. Este mundo, que es agnóstico a las aplicaciones, representa el pilar de conocimiento esencial para un desarrollo exitoso (Lin & Ming, 2021).

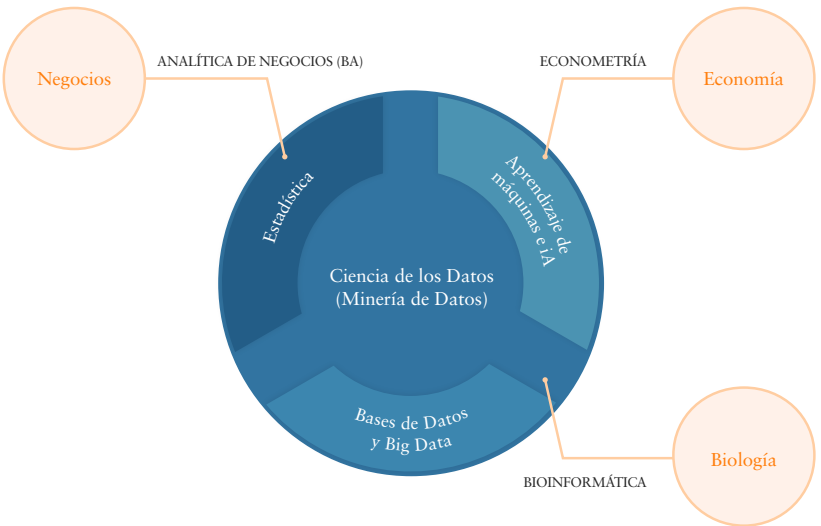
Se distinguen tres ejes o familias de técnicas principales: estadística, inteligencia artificial (IA) y aprendizaje automático, y bases de datos, *Data Warehousing* (DW) y Big Data. A su vez, estos ejes contienen elementos propios, los que serán explorados con mayor profundidad en diversos capítulos del libro. Esta trilogía de estrategias surge como concepto en los años noventa bajo el nombre de **minería de datos**, que corresponde al proceso de descubrir patrones en grandes volúmenes de datos. El concepto minería de datos es una analogía que compara la generación de conocimiento con el proceso de extracción de algún mineral precioso, como oro o diamantes, dentro de una mina (analogía a los datos).

La minería de datos, como concepto, se encuentra algo «pasada de moda», siendo reemplazada por «ciencia de datos» (*data science*, o DS). La ciencia de los datos engloba esencialmente las mismas técnicas, pero se populariza con la inclusión de nuevas tecnologías y tendencias, tales como herramientas de Big Data, manejo de datos no estructurados y aprendizaje profundo o *deep learning*. De este concepto nace el científico de datos o *data scientist*, que corresponde

al analista capaz de dominar estas técnicas tanto conceptualmente, como de forma práctica, liderando su implementación. Este perfil es un híbrido entre un *hacker* de datos (alguien que programa y sabe implementar modelos, pero sin tener conocimientos sólidos en la teoría), un estadístico/matemático y un buen comunicador y tomador de decisiones. El trabajo del científico de datos es reconocido como el trabajo más *sexy* del siglo xxi, debido a que esta combinación es extremadamente poco común, pero, a la vez, muy poderosa y valorada en las empresas (Drucker, 2012).

Los tres ejes asociados a las técnicas y su relación con las aplicaciones se presentan en la figura 1.2.

Figura 1.2: Elementos de la ciencia de los datos




## *1.2.1 Estadística*

En primer lugar, se encuentra todo lo relativo al dominio de conocimientos estadísticos, en adición a una sólida formación en diversas técnicas y herramientas de análisis descriptivo de datos. Utilizando estos conocimientos, es posible comprender la estructura y los patrones que subyacen a los datos, realizar inferencias a partir de estos últimos (Securonix Labs, 2017) y proponer diversas hipótesis utilizando una gran cantidad de técnicas de modelamiento de datos.

Por ejemplo, si nos encontráramos analizando transacciones de una entidad bancaria, los encargados podrían querer saber si existen comportamientos que pudieran ser catalogados como sospechosos de fraude. Lo anterior requerirá de razonamiento estadístico para poder identificar, por ejemplo, comportamientos normales y anómalos en términos de transacciones diarias en las cuentas. Esto es posible analizando la distribución de datos con medidas estadísticas.

Una vez que se logran identificar valores que indiquen un comportamiento sospechoso, estos deberán ser analizados de manera exhaustiva, para comprobar si es posible catalogarlos como fraude o no. Es esencial una comprobación que no considere exclusivamente información descriptiva, sino también un modelamiento avanzado de los datos e incluso criterios de negocios y juicio de expertos.

Así, el objetivo fundamental de la estadística como pilar para la ciencia de datos es describir y realizar inferencias de manera válida y con alto poder explicativo, cumpliendo con los supuestos subyacentes asociados a las técnicas. Por lo anterior, se requiere una sólida formación en este campo, de manera tal que los análisis realizados tengan como base modelos estadísticos precisos, que logren capturar las inferencias y reflejarlas considerando el dinamismo de los datos.

### *1.2.2 Inteligencia artificial y aprendizaje automático*

La característica esencial de este segundo pilar radica en que domina la predicción. Mientras el valor que la estadística le aporta a la ciencia de datos es la interpretación y la comprobación de hipótesis, la IA se enfoca de forma más ciega en aprender sobre los datos y realizar predicciones sin suposiciones explícitas sobre la estructura de los datos (Hale, 2020).

En el aprendizaje de máquinas y la IA, es posible realizar predicciones utilizando las características más importantes de los datos para estos efectos, capturar relaciones complejas en los datos y comprobar empíricamente si las hipótesis realizadas con estadística se mantienen, o no. Esto se realiza con un enfoque de «caja negra», centrándose en la capacidad predictiva por sobre la interpretabilidad del modelo.

Para lograrlo, es esencial tener conocimientos en programación, complementados con los conocimientos estrictamente teóricos de aprendizaje de máquinas y optimización. También es muy importante contar con experiencia en gestión de datos y aplicación de modelos predictivos, ya que existen bastantes riesgos de aplicar los modelos de forma incorrecta y cometer errores como el «sobreajuste». Este concepto será detallado en el capítulo dedicado a la modelación predictiva. Independiente de la estrategia utilizada para predecir, la preocupación principal se concentra en la evaluación de la calidad de la predicción y la minimización de errores.

En resumen, el pilar relativo al aprendizaje automático es parte de los conocimientos más esenciales para lograr de manera exitosa un proceso de ciencia de datos. Conocer qué algoritmos son más adecuados para un problema de negocios, saber cómo modelarlos, cómo evaluarlos e interpretarlos, será fundamental para obtener resultados concretos y que entreguen valor a quienes lo requieran.

### *1.2.3 Bases de datos, almacenamiento y Big Data*

Se estima que cerca de un 80% del tiempo total dedicado al desarrollo de modelos corresponde al proceso de *data crunching*, es decir, limpieza, preparación, procesamiento y conversión de datos en bruto a conjuntos de datos listos para ser analizados (IBM, 2020). Gran parte de este trabajo forma parte del tercer pilar de conocimientos y técnicas fundamentales para el desarrollo de un proceso analítico de datos exitoso. En los últimos años se ha visto un incremento en la necesidad de contar con ingenieros de datos (Ávila, 2019), un cargo separado de las labores del análisis de datos como tal. Sin embargo, actualmente en muchas organizaciones (sobre todo aquellas de menor tamaño y complejidad) esta responsabilidad, y aquella ligada al aprendizaje automático y ciencia de datos como tal, puede recaer en una misma persona.

¿Qué quiere decir lo anterior? Básicamente, que un científico de datos también puede recopilar, gestionar, almacenar y limpiar datos, solo que no es su función primaria (Hale, 2020). Pese a lo anterior,

para poder realizar de manera exitosa un análisis exhaustivo de un problema de negocios se requiere tener nociones o conocimientos de informática, bases de datos y almacenamiento de estos.

El entorno de las bases de datos, la computación e implementación de modelos en plataformas de Big Data plantean un gran desafío para aquellos científicos de datos que no tienen amplia experiencia en informática o programación. En favor de estos últimos, hoy existen herramientas de acceso casi universal, que hacen más sencilla la gestión de grandes fuentes de información, donde se destaca la Big Data en la nube, a través de plataformas como Google Cloud Data Platform, Amazon Web Services, Microsoft Azure, entre otras.

A modo de ejemplo, un científico de datos debería manejar conceptos como *datasets* o conjunto de datos, ETL (extracción, transformación y carga) y también lenguajes como SQL, el cual permite gestionar y manipular modelos de datos relacionales. ¿Para qué le servirían estos conocimientos? Para familiarizarse con los datos; detectar problemas en la base, como los típicos *missing values* o valores incongruentes que **finalmente afectan la calidad de los datos llevados a análisis** (Sayad, 2013); descubrir de manera superficial algunos patrones o subestructuras subyacentes y, obviamente, para tener una comprensión más global de estos.

Cabe destacar que, si bien los tres pilares mencionados son parte de lo fundamental para realizar ciencia de datos de manera exitosa, esto no excluye otro tipo de conocimientos que pueden servir de apoyo para cada uno de los procesos necesarios a llevar a cabo. No es raro que los equipos de trabajo de ciencia de datos estén compuestos por más de dos personas, por lo que las habilidades de comunicación y liderazgo durante todo el trayecto recorrido son fundamentales para internalizar en el equipo los objetivos, la nomenclatura interna y otros aspectos relevantes.

## 1.3 Dimensión práctica: vínculo con aplicaciones y analítica de negocios

Una parte esencial del entrenamiento de un científico de datos es conocer el dominio de aplicación de las técnicas. No basta con

dominar los modelos de forma teórica y ser capaz de programarlos e implementarlos: un científico de datos completo debe ser capaz de manejar el vocabulario y comprender las complejidades de las diferentes áreas de aplicación.

Este libro se enfoca en el vínculo entre el mundo de las técnicas y los negocios como área de aplicación. Como se ilustra en la figura 1.2, este vínculo se conoce como analítica de negocios (*business analytics* o BA). Las fuentes principales de datos son los clientes (para perfilamiento y predicción de comportamiento) y las ventas (para proyección de demanda y decisiones táctico-estratégicas). Sin embargo, estas fuentes de datos se han visto enriquecidas por una gran cantidad de información en la última década, tales como datos georreferenciados, datos textuales (reclamos y comentarios), imágenes, etcétera. Este tipo de fuentes de información representa un desafío mayor desde el punto de vista técnico, ya que suele tener una estructura más difícil de trabajar que los datos tabulares tradicionales.

El vínculo entre el mundo de las técnicas y los negocios nace en los años noventa bajo el concepto de inteligencia de negocios (*business intelligence* o BI). Este surgimiento ocurre con el nacimiento de las técnicas analíticas para el apoyo de la toma de decisiones, como los *data warehouses* y cubos OLAP (procesamiento analítico en línea), además de la popularización de los modelos predictivos en empresas. Al igual que el concepto de minería de datos, el término BI está «pasando de moda» y siendo reemplazado por **analítica de negocios.** Este concepto va más allá que el BI tradicional, que tiene por foco establecer alarmas y realizar consultas y reportes con el fin de dejar al analista la toma de decisiones. En este sentido, la analítica prescriptiva, la automatización y la optimización de las decisiones son los principales elementos diferenciadores del BA con respecto al BI.

Inferir el comportamiento de los clientes no es una tarea fácil. Si bien es cierto que los datos tradicionales de clientes no representan una gran dificultad para la aplicación de modelos desde el punto de vista técnico, la integración de diferentes fuentes (que es la nueva tendencia en negocios) sí lo es. Adicionalmente, existen otros desafíos de la aplicación, tales como el comportamiento más bien errático y a veces hasta «impredecible» de los clientes, la falta de información fidedigna y libre de errores, y los posibles sesgos en los datos.

Es importante destacar que existen diversos vínculos entre el mundo de las técnicas y otras aplicaciones. Por ejemplo, este mundo aplicado a economía se conoce como **econometría**, mientras que la vinculación entre las técnicas y las aplicaciones en biología se conoce como **bioinformática**, entre otros vínculos.

## 1.4 Proceso de extracción de conocimiento

El proceso de minería de datos se basa fundamentalmente en la extracción y explotación de grandes volúmenes de información para la generación de conocimiento dentro de las organizaciones. Los esfuerzos en esta área del conocimiento se han abocado a la investigación de diversas técnicas y algoritmos para el procesamiento de la información y posterior análisis y extracción de patrones en ella. Por el contrario, existe una menor profundización en cómo ejecutar este proceso, es decir, en las metodologías a aplicar para lograr obtener el «nuevo conocimiento», en cómo llevar a cabo las fases, actividades y tareas que involucran el proceso completo.

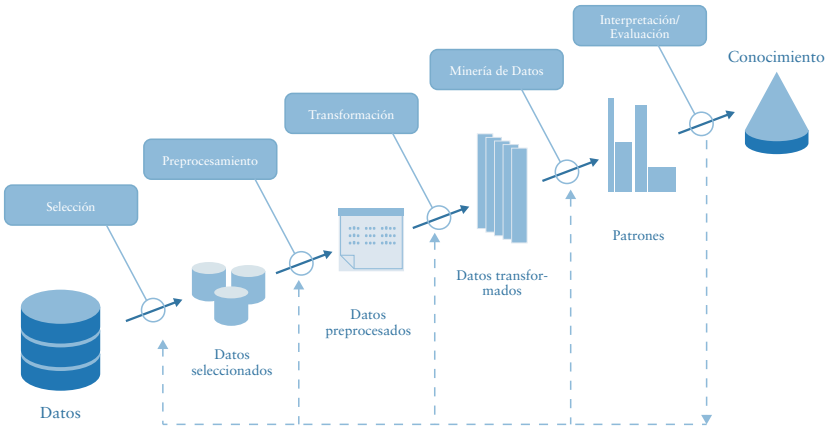
En los inicios de 1996, el modelo KDD *(Knowledge Discovery in Databases)* fue el primer modelo aceptado por la comunidad científica que estableció etapas para un proyecto de minería de datos exitoso. En él se establece que la minería de datos es una de las etapas dentro del proceso definido, en la cual se identifican patrones y tendencias en base a los datos. Actualmente, ambos conceptos (KDD y minería de datos) se utilizan para hacer referencia al proceso completo de descubrimiento de conocimiento. El proceso KDD se define como «El proceso no trivial de identificar válidos, novedosos,

potencialmente útiles y entendibles patrones contenidos en datos» (Fayyad *et al.*, 1996).

Dentro de esta definición, el modelo KDD se puede definir como un proceso que se compone de diversos pasos, entre los cuales se encuentran la selección, la preparación de los datos, la búsqueda de patrones y tendencias, la posterior evaluación y perfeccionamiento de los modelos, además del conocimiento en múltiples ocasiones. Por otra parte, se dice que es un proceso no trivial, ya que requiere de investigación, búsqueda e inferencia para lograr a llegar a resultados interpretables y útiles.

El proceso KDD se presenta en la figura 1.3. Cuenta con los siguientes pasos:

Figura 1.3: Proceso KDD




1. **Selección de un conjunto de datos** o un subconjunto de ellos, en base a los cuales se realizan análisis y descubrimientos.
2. **Preprocesamiento y limpieza de datos**, decidiendo distintas estrategias para manejar campos vacíos, valores inválidos y valores que ensucien el conjunto de datos a analizar, tales como los valores extremos, mal formateados, etcétera.
3. **Transformación de datos**, dependiendo de los requerimientos del objetivo planteado. Es recomendable, dependiendo del problema, normalizar o escalar valores numéricos, generar nuevas variables que puedan servir para el modelamiento

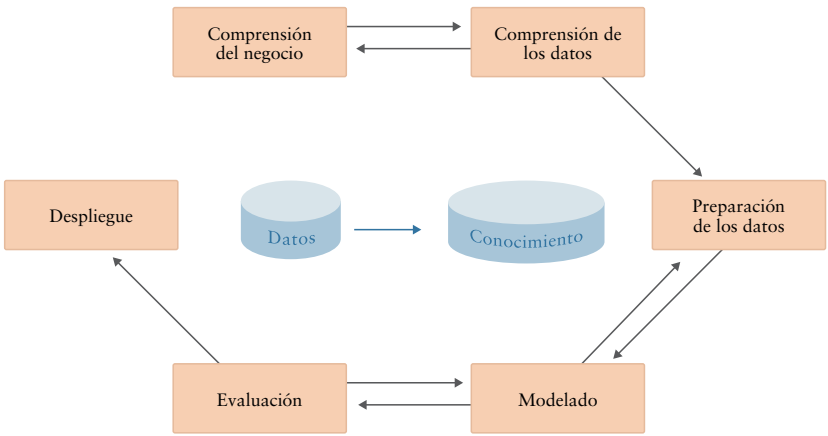
u otras maneras de alterar la data. Además, es necesario simplificar los conjuntos de datos determinando cuáles son las variables de interés.

4. **Modelado**, etapa en la cual se aplica el *data mining* como tal con mayor profundidad, donde se selecciona uno o más técnicas que sean apropiadas y que cumplan los requisitos para resolver el problema, ya sean de clasificación, regresión, reglas de asociación u otros. Esta etapa incluye la **selección** de la o las técnicas a aplicar, con todos los supuestos y características específicas de cada una; **pruebas de calidad** de los modelos a aplicar, en las cuales se hace necesaria la validación cruzada, separando los conjuntos de datos en uno de entrenamiento y otro de prueba; la **construcción** del modelo definitivo para el conjunto de datos preparado, seleccionando de manera iterativa los parámetros específicos bajo los cuales se aplicará, y finalmente, la interpretación del modelo de acuerdo a los criterios establecidos de manera previa.
5. **Interpretación/evaluación** del o los modelos, basándose en los criterios de éxito establecidos. Aquí, tanto el proceso completo como los resultados deben ser interpretados y evaluados, de manera tal que se logre alcanzar una calidad óptima de los resultados obtenidos, haciendo posible extraer conocimiento concreto que sea transformable en posibles acciones para atacar el problema de negocios.

Por otra parte, se encuentra la metodología CRISP-DM (abreviación de *cross-industry standard process for data mining*), la cual es una adaptación del proceso KDD con pasos más cercanos a la industria (Chapman *et al.*, 2000). Esta metodología proviene de las empresas Daimler Chrysler y SPSS, pioneras por su dedicación de funciones basadas en la explotación de información y técnicas de minería de datos. CRISP-DM es una metodología basada en la práctica y experiencia real de analistas de minería de datos, la cual se organiza en fases, procesos, documentos entregables y actividades, tal como se ilustra en la figura 1.4. Es relevante entonces mencionar

que, dentro de cada una de las fases próximas a mencionar, se deberán realizar diversos procesos y actividades para el cumplimiento del proceso completo.

Figura 1.4: Proceso CRISP-DM




- **Comprensión del negocio:** el propósito de esta fase es conocer una perspectiva clara y realista del negocio, de sus objetivos, sus recursos actuales (a nivel de fuentes de datos, *software*, *hardware* y personal), sus problemas y preguntas a responder, de manera tal que los esfuerzos a realizar vayan en la dirección correcta.
  Para lo anterior y considerando principalmente los objetivos del negocio, es necesario listar tanto un objetivo principal como objetivos secundarios que contribuyan a su consecución, los cuales deben ser medibles en su éxito, alcanzables y concretos. Por otra parte, conociendo los objetivos y recursos presentes en la situación actual del negocio, se deben plantear requerimientos, supuestos y limitaciones (presupuestarias y/o legales) para la realización del proyecto a proponer, al mismo tiempo que se identifiquen sus riesgos, ya sea por problemas de negocio, limitaciones económicas, tecnológicas o baja calidad de los datos disponibles.
  Conociendo todos estos aspectos, tanto internos como externos, del negocio, es que se debe definir el problema de minería de

datos que será abordado y el objetivo planteado para resolverlo, elaborando una planificación formal preliminar para el cumplimiento del objetivo principal del negocio.

- **Comprensión de los datos:** esta fase considera la colección de los datos en su estado inicial, además de guardar relación con el trabajo dirigido a familiarizarse de la mejor manera posible con estos, conocer su origen, calidad, características, estructura, problemas, riesgos, etcétera.

  Debido a que los datos son la materia esencial para la minería de datos, esta fase es fundamental para lograr una calidad de datos lo más alta posible; condición necesaria, pero no suficiente, para un proceso de minería de datos exitoso.
- **Preparación de los datos:** la fase de preparación considera todas las actividades requeridas para conseguir el conjunto de datos final, sobre el cual se aplicarán los modelos y herramientas.

  Entre las actividades requeridas se incluye la **selección** de datos, utilizando criterios para la selección y exclusión de atributos y datos; **limpieza**, de manera tal que se garantice la calidad de datos necesaria; **transformación**, generando nuevos atributos o variables, formateando, normalizando o escalando valores numéricos, entre otras, e **integración** en los casos que aplique la utilización de distintas fuentes de información.
- **Modelado:** el propósito fundamental de la fase de modelado radica en disponer de un modelo que dé soporte a la consecución del o los objetivos de minería de datos, también de los objetivos del negocio ya establecidos previamente. Es en esta fase donde se aplican diversas técnicas (mientras más técnicas a aplicar, mejor) al conjunto de datos final construido en la fase de **preparación**, con el objetivo de identificar patrones, o bien predecir el comportamiento de una variable objetivo.

  La selección de las técnicas de modelado a aplicar va a depender netamente del problema de minería de datos definido, el cual puede ser de segmentación, clasificación, predicción u otro tipo de problema. Para cada uno existirá una o varias técnicas que sean aplicables en cada caso.

Cabe destacar que esta fase es normalmente iterativa tanto con su fase previa, como con la posterior, debido a la existencia de técnicas o algoritmos que poseen requerimientos relativos a la estructura o formato de los datos.

- **Evaluación del modelo:** de manera previa a la última fase de implementación, es sumamente relevante evaluar la calidad tanto de los resultados en base a criterios de éxito, como del acercamiento de los objetivos del negocio a ellos, es decir, si logra responder las preguntas planteadas en un principio o si logra contribuir de manera directa o indirecta a los objetivos. Por otra parte, será necesario revisar todo el proceso de minería de datos en los modelos que hayan sido aprobados en la revisión anterior, para identificar errores, propuestas de mejora y optimización en tareas y/o actividades específicas.
- **Despliegue o implementación:** una vez determinados los modelos aprobados y tomadas las conclusiones más relevantes, estas deben ser aplicadas en los niveles organizativos y partes interesadas planteadas en las etapas iniciales. En esta etapa se define cómo los conocimientos serán propagados hacia las partes interesadas, realizar seguimiento y constante revisión del despliegue, incluyendo entrevistas a usuarios finales y autoevaluaciones para identificar posibles aspectos a mejorar en la metodología aplicada.

Dentro del mismo marco del proceso de extracción de conocimiento, es necesario describir las diferencias existentes entre las técnicas de aprendizaje supervisado y no supervisado, las cuales forman parte de etapas claves de cualquier proceso de minería de datos. A partir de la definición del problema de negocios, es necesario establecer qué tipo de aprendizaje automático será realizado, considerando el contexto o escenario del problema, también la estructura del conjunto de datos a utilizar.

En el **aprendizaje automático supervisado**, los modelos matemáticos serán construidos o «entrenados» utilizando datos que ya tienen una etiqueta o valor predeterminado, tanto de entrada como de salida. Lo anterior quiere decir, en este caso, que el algoritmo conoce de

manera previa el resultado y debe reconocer los patrones al interior del conjunto de datos, para que en etapas posteriores sea capaz de predecir el resultado con nuevos datos. Esta técnica de aprendizaje puede utilizarse en problemas de regresión o de clasificación.

Por ejemplo, suponiendo un conjunto de datos conformado por imágenes de animales, la tarea del modelo de aprendizaje supervisado será identificar los tipos de animales y clasificarlos según corresponda (aves, reptiles, anfibios, etcétera). Para esta tarea se proporcionarán como datos de entrada imágenes con animales claramente identificados (con sus respectivas etiquetas), de manera que este entrene por la forma, tamaño, colores y características específicas de las imágenes. Una vez finalizada la etapa de entrenamiento, se probará el modelo con un conjunto nuevo de datos con imágenes nuevas sin etiquetas, para que este pueda identificar el tipo de animal al que corresponde la imagen y clasificarlo, prediciendo el resultado de lo aprendido con el conjunto anterior.

Por otro lado, en el **aprendizaje automático no supervisado**, el modelo debe entrenar e inferir patrones dentro de un conjunto de datos de entrada, cuyo valor o etiqueta no está determinado. Es un tipo de aprendizaje que no requiere supervisión, puesto que encuentra patrones dentro de la estructura del conjunto de datos por sí solo, por lo que puede asociarse de manera más cercana a la verdadera inteligencia artificial, ya que se trata de un modelo que aprende de manera iterativa sobre datos sin el conocimiento de su valor final. Puede utilizarse en problemas de asociación y agrupación o *clustering*.

A modo de ejemplo y comparación con el caso anterior, suponemos el mismo conjunto de datos, pero esta vez sin los datos de salida. Estos datos serán proporcionados al modelo, de manera que este entrene y encuentre los patrones y estructuras internas de los datos para identificar el tipo de animal respectivo y clasificar el dato según el entrenamiento y aprendizaje logrado.

## 1.5 Dimensión práctica: lenguaje de programación Python

Tal como se mencionó en el prefacio, en este libro nos «ensuciaremos las manos» utilizando códigos de lenguaje Python para la aplicación de diversas técnicas y metodologías analíticas. Del mismo modo, en la parte inicial de esta sección se hizo mención a la constante evolución de las distintas herramientas y *softwares* que son utilizadas como apoyo y base de las diversas prácticas analíticas, donde Python corresponde a un claro ejemplo de la evolución y enorme utilidad en analítica y ciencia de datos. Consideramos lo anterior como una de las principales razones por las cuales se utilizará el lenguaje Python para las aplicaciones a realizar en el libro.

Contamos hoy con diversos *softwares* y lenguajes de programación para realizar prácticas analíticas. La primera distinción para realizar es entre aquellos de código abierto (Python, R, Rapidminer, Weka), es decir, que no tienen ningún costo asociado; y los comerciales (SAS, SPSS, MatLab, MS). Muchos de estos proveedores de *software* se enfocan en generar y vender soluciones analíticas para problemas específicos a nivel de organización, departamento o funciones.

Ya en 2012 o 2013, Python se encontraba dentro de los lenguajes de programación más utilizados en prácticas analíticas, según una encuesta realizada por KDNuggets (Piatetsky, 2019). Dicha posición en las preferencias de los usuarios ha variado positivamente, dado que desde el 2017 se encuentra en el primer lugar de los rankings, llegando a obtener un 86% de preferencias en la encuesta realizada en 2020 por Kaggle ML. En estudios más recientes, como el publicado el 2020 por Paul Mooney, Python se encontraba en la primera preferencia no solo en popularidad, sino también en la recomendación del primer lenguaje a aprender para utilizar en ciencia de datos (Mooney, 2020).

Tal como se mencionó anteriormente, Python es un *software* de código abierto, pero además es un lenguaje potente y fácil de usar para comenzar a trabajar en diversos tipos de proyectos, ya seas estudiante con conocimientos básicos o un investigador experto

(Hebbar, 2019). Por otra parte, Python proporciona una gran diversidad de librerías para realizar aprendizaje automático o inteligencia artificial, tales como *Numpy*, *Pandas*, *Matplotlib*, *Scikit-learn* o *Scipy*, las cuales servirán para muchas de las prácticas que serán descritas en los próximos capítulos del libro. Finalmente, y no por eso menos importante, es un lenguaje que posee una comunidad muy activa dentro de la disciplina de ciencia de datos y analítica, en la cual muchos usuarios voluntariamente crean librerías de primer nivel para problemas de ciencia de datos, por lo que existe un constante desarrollo de herramientas y recursos útiles para diversos tipos de aplicaciones (Kundariya, 2021).

Se mencionó a Python como un ejemplo de la constante evolución de las herramientas utilizadas en la ciencia de datos y prácticas analíticas, dado que es un lenguaje de programación que se desarrolla de manera continua, actualizando sus recursos para ser más eficiente en tareas a realizar, mejorando su disponibilidad y acceso, a través de la facilidad de su uso, entre otras razones.

Por las razones descritas anteriormente, creemos que la utilización de Python como nuestra caja de herramientas a lo largo del libro y del desarrollo de los casos de aplicación ayudará al lector a identificar y entender de manera sencilla, amena y explícita, cada uno de los conceptos involucrados en los modelos propuestos, los errores más comunes de cometer y los patrones presentes en los distintos casos de aplicación a revisar.

## Referencias

Amor, E. (2020). *How to manage complexity and realize the value of* Big Data. Recuperado de: https://www.ibm.com/blogs/services/2020/05/28/how-to-manage-complexity-and-realize-the-value-of-big-data/

Ávila, M. (2019). *¿Cuál es la diferencia entre un ingeniero de datos, un analista de datos y un científico de datos?* Recuperado de: https://www.linkedin.com/pulse/cu%C3%A1l-es-la-diferencia-entre-un-ingeniero-de-datos-analista-avila/?originalSubdomain=es

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. & Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium.

Drucker, P. (2012). «Data Scientist: The Sexiest Job of the 21st Century». *Harvard Business Review,* recuperado de https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century.

Fayyad, U. (1996). «Data mining and knowledge discovery- making sense out of data». *IEEE Expert,* 11(5), 20-25.

Hale, J. (2020). *What is Data Science?* Recuperado de: https://towardsdatascience.com/what-is-data-science-8c8fbaef1d37

Hebbar, P. (2019). *Why Should You Learn Python For Data Science?* Recuperado de: https://www.esparkinfo.com/python-for-data-science.html

Hitner, N. (2017). *The Four Types of Data Analytics*. Recuperado de: https://datafloq.com/read/the-four-types-of-data-analytics/3903

IBM (2020). *Data Preparation*. Recuperado de: https://www.ibm.com/cl-es/analytics/data-preparation

Kundariya, H. (2021). *Why Python is Essential for Data Science?* Recuperado de: https://www.esparkinfo.com/python-for-data-science.html

Lin, H. & Ming, L. (2021). *Introduction to Data Science*. Recuperado de: https://scientistcafe.com/ids/three-pillars-of-knowledge.html

Mooney, P. (2020). *2020 Kaggle Data Science & Machine Learning Survey*. Recuperado de: https://www.kaggle.com/paultimothymooney/2020-kaggle-data-science-machine-learning

Piatetsky, G. (2019). *Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis*. Recuperado de: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

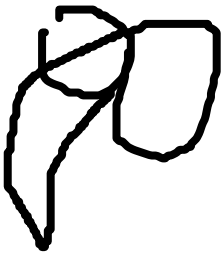
Sapp, C., Brabham, D., Antelmi, J., Cook, H., Craig, T., Barot, S., Galli, D., Pal, S., Mohan, S. & Gilbert, G. (2019). *2019 Planning Guide for Data and Analytics*. Recuperado de: https://www.gartner.com/en/doc/361501-2019-planning-guide-for-data-and-analytics

Sayad, S. (2013). *Data Mining Map*. Recuperado de: http://www.saedsayad.com/data_mining_map.html

Securonix Labs (2017). *Data Science: Statistics vs. Machine Learning*. Recuperado de: https://www.securonix.com/ch-2-data-science-statistics-vs-machine-learning/

Zhidkov, R. (2020). *Why Python is Essential for Data Analysis*. Recuperado de: https://www.rtinsights.com/why-python-is-essential-for-data-analysis/

# 2. *Descriptive Analytics* y preprocesamiento



Desde que los humanos nacen se ven influenciados por diversas situaciones en las cuales deben tomar decisiones que les permitan desarrollarse e interactuar con las demás especies. Cuando nos enfrentamos por primera vez a un escenario en el cual no tenemos certeza de cómo actuar, se suele tomar una decisión y posteriormente se evalúa si esta fue o no la correcta. Pero cuando ya se ha enfrentado a esta situación varias veces, los individuos ya saben cómo actuar, pues tienen nociones de qué es, para ellos, lo correcto o preferido y, por ende, basado en experiencias anteriores seleccionan el mejor camino para enfrentar los escenarios.

Estas decisiones no serían posibles si los humanos no recordásemos el resultado que se ha obtenido, con las decisiones del mismo escenario, en oportunidades anteriores. Por ende, el poder contar con la información histórica del comportamiento permite a los individuos tomar mejores decisiones.

Este mismo razonamiento se puede aplicar a empresas, ya que estas han debido enfrentar escenarios en los cuales la toma de decisiones ha implicado importantes cambios de estrategia y operación. A diferencia de los humanos, las organizaciones por sí solas no cuentan con la «memoria» que les permita recordar el efecto de sus decisiones al enfrentarse a los diversos escenarios. Por esta razón, nos volcamos en la historia que nos puedan contar los datos.

Para que la toma de decisiones pueda tener un respaldo sólido y argumentos concretos, debemos procesarlos de tal forma que podamos escuchar la historia que ellos nos pueden contar. El primer estudio que podemos realizar es el llamado **análisis exploratorio de**

**los datos** (AED). A partir de este análisis podemos identificar los primeros patrones o «historias» que nos permiten entender el pasado.

AED corresponde al *descriptive analytics* más tradicional, basado en estadística clásica y gráficos simples. No podemos subestimar la importancia de estas técnicas, que de ninguna manera han sido reemplazadas por la inteligencia artificial. La estadística descriptiva se encuentra tan vigente como siempre, y sigue siendo el primer paso recomendado para tener un primer acercamiento a los datos (Baesens, 2014; James *et al*., 2017).

AED, sin embargo, tiene un segundo objetivo: identificar problemas. Las bases de datos suelen estar «sucias» en el sentido de que existe información faltante, inconsistencias, datos que no tienen sentido matemático y problemas en la consolidación y combinación de datos. Este problema suele ser más común con datos de empresas que con otro tipo de datos.

Para realizar un correcto trabajo en las bases de datos, primero debemos someterlas a una etapa de **preprocesamiento** de la información. En esta etapa se analizan todas las variables involucradas para revisar si existen inconsistencias o si falta información (valores perdidos o *missing values*), para poder efectuar las modificaciones adecuadas y, entonces, contar con una base de datos robusta y sólida para aplicar técnicas estadísticas posteriores, alcanzando un mejor análisis de la información.

Se estima que las etapas previas a la construcción de modelos consumen cerca del 70 u 80% del tiempo de un analista. Por ejemplo, la encuesta realizada por la consultora Anaconda[1] llamada «El estado de la ciencia de los datos 2020», sugiere que el 66% del tiempo el científico de datos se encuentra realizando las tareas de carga de datos (19%), limpieza (26%) y visualización (21%). Este porcentaje sería mucho mayor si nos limitamos a problemas de negocios con datos de clientes.

[1] https://www.anaconda.com/state-of-data-science-2020

Una de las principales razones por las cuales toma tanto tiempo preprocesar datos de negocios es que estos no fueron almacenados con propósitos analíticos. Pensémoslo un minuto, ¿por qué bancos, *retail*, empresas de telecomunicaciones, etcétera, almacenan datos? El objetivo primario no es para crear perfiles de clientes o realizar pronósticos de comportamiento o ventas, sino razones netamente transaccionales: no perder el dinero de los cuentacorrentistas, saber cuánto debe un cliente que recibió un crédito y generar los detalles de facturación de los clientes.

La información que es relevante para estas tareas transaccionales no es la misma que para tareas analíticas. Esto lleva a las empresas a descuidar datos importantes para la modelación predictiva. Por ejemplo, la dirección del cliente es uno de los datos transaccionales más importantes para una empresa, ya que debo saber exactamente adónde enviar la facturación. Sin embargo, es un pésimo dato analítico, ya que no permite generar tendencias ni perfiles. Por otro lado, la edad es un excelente atributo para esto, pero tiene escaso valor transaccional.

Lo positivo es que las empresas almacenan datos, es decir, ya tenemos un punto de partida y no necesitamos adquirirlo mediante procesos costosos. Sin embargo, estos datos se encuentran disponibles con otro propósito.

## 2.1 Análisis exploratorio de datos (AED)

Como se mencionó anteriormente, el objetivo principal del AED es doble: (1) encontrar los primeros patrones y tendencias en los datos y (2) identificar problemas en ellos. La primera tarea se considera como *descriptive analytics*, mientras que la segunda corresponde al «paso cero» del preprocesamiento, y nos permite diseñar una estrategia de corrección de errores para preparar nuestros datos para la modelación predictiva.

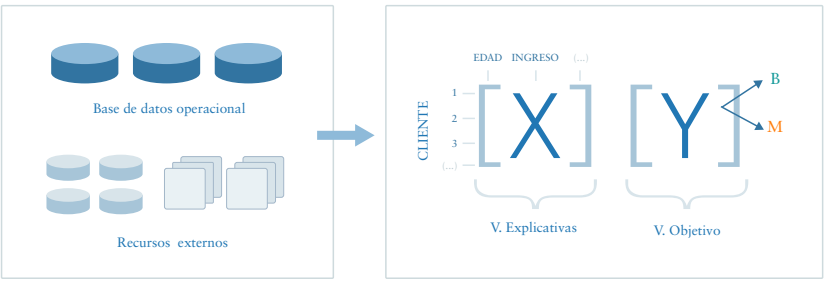
El AED es una etapa muy importante al momento de analizar la información, pues permite organizar, visualizar, preparar datos,

detectar fallos en las bases, identificar los casos atípicos y comprobar los supuestos subyacentes a las técnicas utilizadas, entre otros beneficios que otorga el hecho de conocer en profundidad las bases de datos, su composición y comportamiento.

Primero formalizaremos la estructura de los datos: comenzando con datos transaccionales, tales como las bases relacionales que dispone la empresa y otros datos externos, vamos a combinarlas y juntar una serie de ejemplos u observaciones (clientes, por ejemplo), que serán las **filas** de una matriz X. Cada ejemplo se encuentra caracterizado por una serie de variables o atributos (edad o ingreso, por ejemplo), las cuales conforman las **columnas** de la matriz X.

Además de la matriz X, llamaremos Y a una columna única que será nuestra **variable objetivo** en una tarea predictiva. En un problema de otorgamiento de crédito, por ejemplo, queremos predecir si un cliente que recibe un crédito será un buen (B) o un mal (M) pagador. Cada cliente de nuestros registros históricos tendrá valores en las variables dentro de la matriz X (que llamaremos **explicativas**) y un valor en la variable objetivo (Thomas, 2009). Esto se ilustra en la figura 2.1.

Figura 2.1: Caracterización de la información




Las variables se miden en diferentes escalas: las **cualitativas o nominales** son aquellas no numéricas y que no tienen orden ni distancia entre sus valores o **niveles**. Por ejemplo, la misma variable objetivo anterior es nominal ya que tiene dos valores mutuamente excluyentes («B» o «M»). Otro ejemplo es «estado civil», con cuatro posibles niveles: soltero, casado, divorciado o viudo.

Otro tipo de variables son las **ordinales**, las cuales tienen un orden claro dentro de los niveles, pero las distancias entre ellas no tienen sentido o es difícil de definir. Por ejemplo, una variable con niveles «bueno», «regular» y «malo» presenta estas características.

Finalmente tenemos las variables **numéricas o cuantitativas**, las cuales tienen tanto un orden como una distancia clara. Por ejemplo, el número de personas que componen el núcleo familiar. Claramente, la distancia entre 1 y 2 es la misma que entre 2 y 3. Existen otras subcategorías para las variables numéricas (escala absoluta, discreta, continuas, etcétera) pero no entraremos en mayor detalle.

Es extremadamente importante identificar el tipo de variable, pues de esta manera sabremos cómo manipular la información en caso de que sea necesario aplicar modificaciones para conservar la base de datos completa.

Si bien las variables ordinales son poco comunes, la diferencia entre nominal y numérica es clave. Dependiendo del tipo de dato se definirá el método para corregir irregularidades y, en casos específicos, eliminar variables que no sumen al modelo. Por ejemplo, si reemplazamos los valores faltantes de una variable nominal por el promedio, estaremos cometiendo un error de principiante muy grave. Pensemos en la variable «género» con dos valores: 0 (masculino) y 1 (femenino). El promedio de esta variable es 0.57. Si falta un dato dentro de esta columna, ¿lo reemplazamos por 0.57? Este valor no tiene sentido ya que no representa ni un hombre ni una mujer. Ya volveremos a este tema.

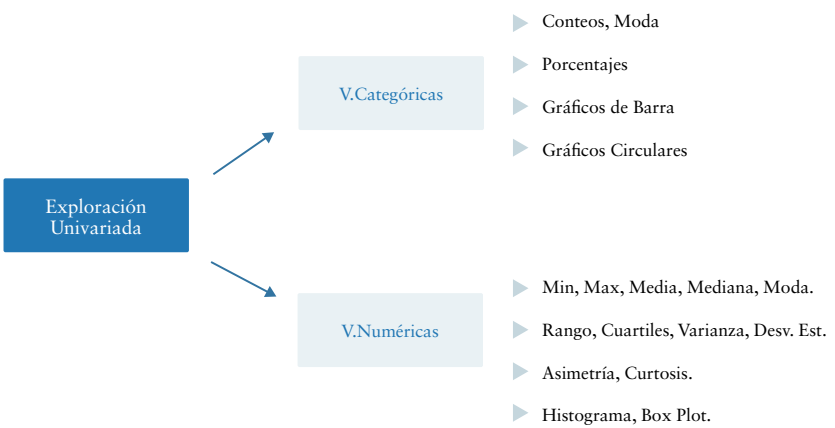
Al momento de querer explorar la data, se pueden emplear diversos métodos o herramientas. Por una parte, podremos contar con tablas y estadísticos descriptivos que detallen la información de las variables y, por otra parte, podemos utilizar gráficos que nos permitirán visualizar de mejor manera la distribución de los datos.

El AED puede ser de dos tipos: **univariado**, es decir, se analiza una variable a la vez, o bien **bivariado**, donde se estudia la relación entre dos variables (en general, una de las columnas de la matriz X con la variable objetivo Y). Ambos tipos se estudian a continuación.

### *2.1.1 Análisis univariado*

Las técnicas para analizar la información de una variable a la vez dependen del tipo de variable. Si las variables son numéricas, tenemos un espectro amplio de estadística descriptiva que podemos considerar (mínimo, máximo, promedio, desviación estándar, cuartiles, etcétera), mientras que este espectro se limita cuando las variables son nominales. Las diferentes técnicas para el análisis univariado se presentan en la figura 2.2.

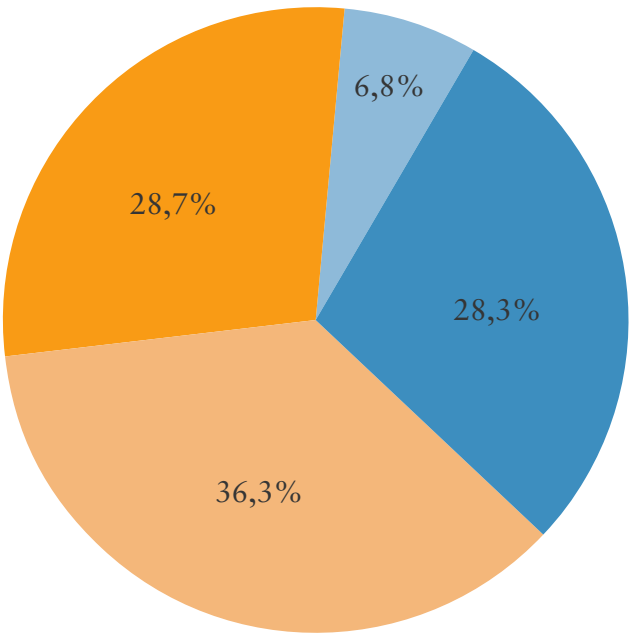
Figura 2.2: Técnicas para el análisis univariado




Volviendo al ejemplo del género, el promedio de la variable no hace sentido, siendo la **moda** (el nivel más frecuente) la única medida de tendencia central razonable. Podríamos reemplazar la información faltante con la moda, por ejemplo. Podemos realizar conteos de categorías e ilustrar estos conteos en un gráfico de barras (figura 2.3). Alternativamente, podemos representar estos niveles en términos relativos al total, es decir, como porcentajes, en un gráfico circular (figura 2.4). Estos gráficos permiten detectar, por ejemplo, categorías poco representadas en los datos, las cuales podrían ser fusionadas con otras con el fin de lograr validez estadística.

Figura 2.3: Gráfico de barras simple

Figura 2.4: Gráfico circular



Con las variables numéricas, en cambio, tenemos más variedad de estadísticos. Como medidas de tendencia central podemos identificar la **media**, la **mediana** (valor del medio en una lista ordenada) y la moda. La mediana puede ser útil para reemplazar información faltante cuando los valores son discretos. Por ejemplo, si falta información en la variable «edad» y la edad promedio es 35.6, entonces puede tener más sentido utilizar la mediana. Esta será un valor dentro del conjunto de datos (por ejemplo, 35 años) y, por ende, tendrá la misma escala que ellos. Alternativamente, siempre podemos redondear la media para mantener su estructura de valores discretos.

Las medidas de dispersión nos indican cómo se comportan los datos en torno a la media. Dentro de estas tenemos el **rango**, la **varianza**, la **desviación estándar** y los **cuartiles**, entre otros.

El rango se obtiene mediante el mínimo y máximo de la variable. Estos valores son extremadamente importantes ya que nos permite identificar inconsistencias en los datos. Es importante considerar que hay dos tipos de valores extremos o *outliers*: inconsistencias y valores reales. Las inconsistencias se encuentran fuera del rango normal de la variable. Por ejemplo, una edad de 1970 es claramente una inconsistencia, mientras que una edad de 77 años puede ser un valor extremo en una base de datos de otorgamiento de crédito, pero es un valor posible, al fin y al cabo.

El mínimo y el máximo permiten contrastar los valores extremos con los valores esperados para una variable. El rango esperado es algo que el científico de datos sabe, pero el computador no. En otras palabras, la identificación de inconsistencias es algo que no se puede automatizar y requiere la participación del analista.

Al considerar calcular la varianza de la muestra, estamos evaluando el promedio de las desviaciones de la muestra con respecto a la media al cuadrado. Formalmente:

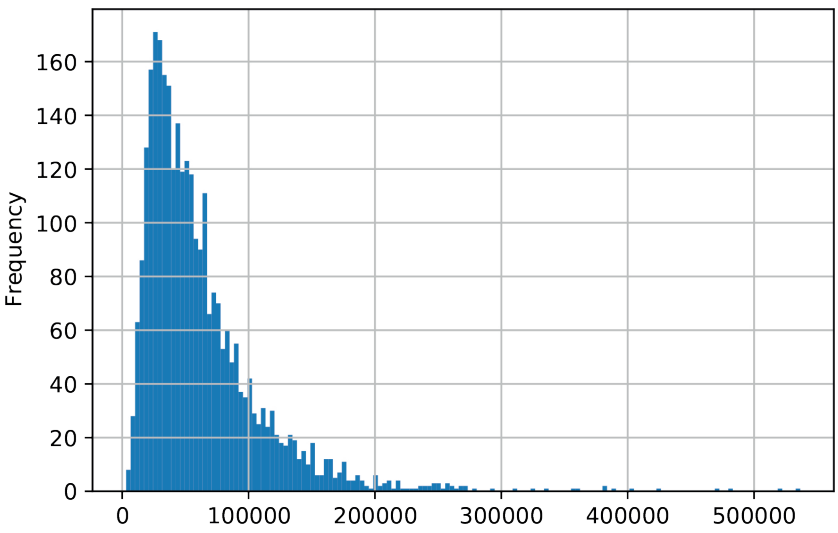
Ecuación 2.1

$$\sigma^2 = \frac{\sum_{i=1}^{N}(X_i - \mu)^2}{N}$$

A su vez, la desviación estándar se calcula como la raíz cuadrada de la varianza. La ventaja de la desviación estándar es que tiene las mismas unidades que la variable original. Es decir, la desviación estándar del ingreso refleja la dispersión de la variable en unidades de dinero.

La distribución de los datos se puede representar mediante un histograma. A diferencia del gráfico de barras, ya no podemos representar la frecuencia de un nivel contando las veces que se repite un valor. En una variable como ingreso, por ejemplo, tendremos una cantidad enorme de valores diferentes. Por esto, agrupamos en intervalos o *bins* los diferentes valores de la variable y contamos la frecuencia de las observaciones en cada intervalo. La figura 2.5 muestra el histograma de la variable «ingreso» para la base de datos de nuestro Laboratorio 1, utilizando 150 *bins*.

Figura 2.5: Histograma para la variable ingreso



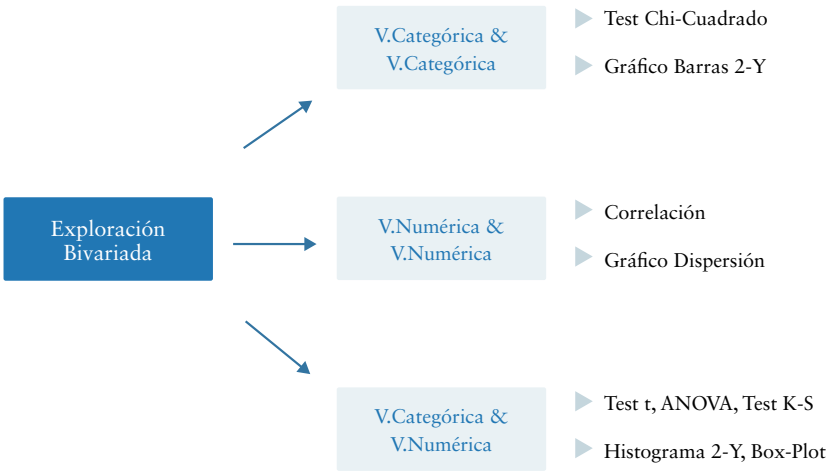
### *2.1.2 Análisis bivariado*

Uno de los grandes factores que distingue a un buen analista de datos son los conocimientos de matemáticas y estadísticas aplicadas que posee, ya que estas forman parte de las bases sobre las que se construyen todos los modelos matemáticos utilizados en *analytics*.

Por esta razón, conocer herramientas como el análisis bivariado (o bivariante) se hace fundamental, tanto para comprender mejor la información disponible de su compañía, como para obtener el mayor rendimiento posible a partir de los modelos de *predictive analytics*.

El análisis bivariado permite estudiar las relaciones existentes entre dos variables, con la finalidad de establecer relaciones causa-efecto (tendencias) producidas entre ellas. Antes de ahondar en los métodos, es importante tener claro que la estrategia bivariada a seguir depende del tipo de variable (a grandes rasgos, si es numérica o categórica), como se muestra en la figura 2.6.

Figura 2.6: Técnicas para el análisis bivariado




#### *2.1.2.1 Análisis de correlaciones*

Una de las técnicas que se ha vuelto indispensable para todo analista de datos es el análisis de las correlaciones, el cual expresa una tasa de cambio constante que define el grado en que dos variables están relacionadas linealmente.

A continuación, se presenta la fórmula para calcular la correlación de Pearson *(rho de Pearson)*, ampliamente utilizada en el contexto de *analytics:*

Ecuación 2.2

$$\rho_P = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Donde *n* corresponde a la cantidad de observaciones para las variables $X_i, Y_i$ con $i \in \{1, \ldots, n\}$. La medida corresponde a la covarianza entre las dos variables X e Y normalizada por la raíz de las respectivas varianzas.

Esta medida permite cuantificar la dependencia lineal entre dos variables entre los valores -1 y +1, donde +1 significa que existe una relación de correlación fuertemente positiva (si una aumenta, la otra también lo hará), mientras que -1 significa que existe una relación de correlación fuertemente negativa (si una aumenta, la otra disminuirá).

Ahora bien, en caso de que la correlación sea 0, se concluye que no existe correlación entre ambas variables, lo cual quiere decir que, si una aumenta o disminuye, entonces la otra no se verá afectada (variables independientes).

Cabe mencionar que el análisis de correlaciones sirve principalmente para reconocer relaciones lineales entre variables, es decir, no permite reconocer patrones no lineales (patrones que tienen un distinto comportamiento dependiendo del rango de valores estudiado).

### *2.1.2.2 Gráficos para análisis bivariado*

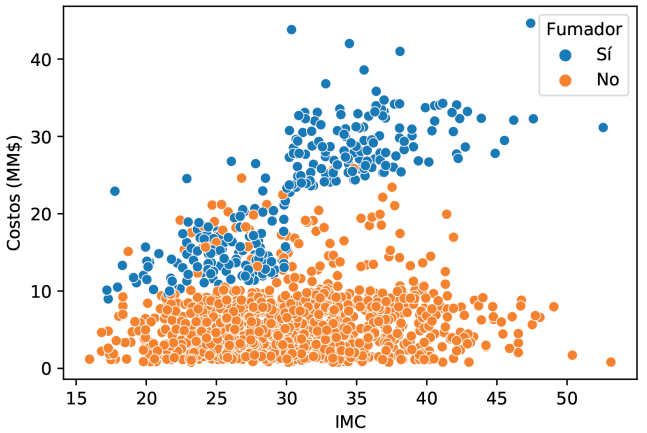
Como mencionamos anteriormente, las visualizaciones gráficas de los datos almacenados en las compañías resultan de gran utilidad ya que permiten obtener información que surge de la relación entre una o más variables. Dentro del contexto del análisis de las correlaciones, existen visualizaciones gráficas que permiten representar de manera didáctica la presencia de correlación entre múltiples variables, de manera que los analistas puedan identificar patrones visualmente.

Por otro lado, el análisis bivariado no solo sirve para observar las correlaciones, sino que también es muy útil para observar, mediante una inspección simple, la distribución de los valores de una variable, también información acerca de estadísticas básicas como moda, media, mediana y los rangos intercuartílicos.

A continuación, se presentan tres gráficos que permiten realizar un análisis bivariado entre múltiples variables:

- ***Scatterplot* o gráfico de dispersión**: una de las mejores alternativas para representar gráficamente la correlación entre dos variables es por medio de un *Scatter plot* o gráfico de dispersión, el cual muestra la relación entre los valores de dos variables en el plano X-Y, mediante una figura determinada (en este caso mediante círculos rellenos, como se muestra en la figura 2.7). Además, se puede incorporar una tercera variable de naturaleza categórica con la finalidad de diferenciar dos o más categorías a las cuales pertenecen las observaciones graficadas.

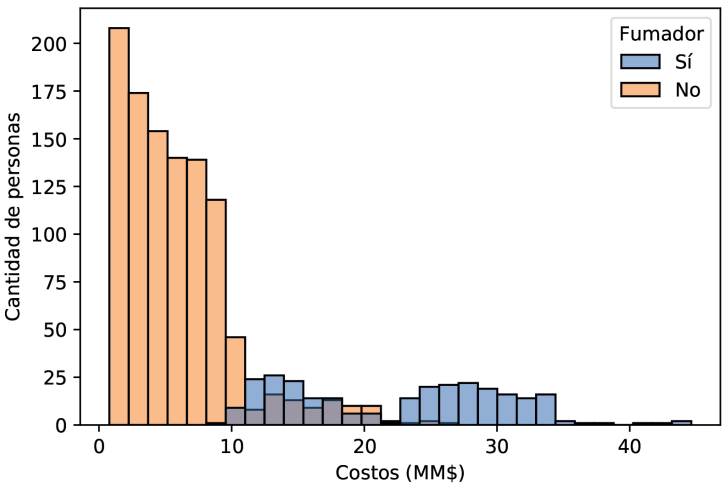
Figura 2.7: Scatter plot – Índice de masa corporal (X) y cargos por aseguramiento (Y)



En la figura 2.7 se presenta un ejemplo de diagrama de dispersión para visualizar la correlación entre las variables índice de masa corporal (IMC) y los costos (o cargos) que debe pagar una aseguradora por cada cliente, dentro del contexto del tabaquismo. Se puede observar que aquellos asegurados que son fumadores poseen un costo sustancialmente más alto para las aseguradoras, en la medida que poseen un grado más alto de sobrepeso (IMC más alto).

- **Histograma 2-Y:** anteriormente fue presentado el histograma dentro de las estrategias más comunes para visualizar la distribución de una variable numérica. Este tipo de gráficos se puede adaptar para superponer dos distribuciones y analizar visualmente las diferencias entre estas. A modo de ejemplo, la figura 2.8 muestra un histograma que contrasta los costos de una aseguradora por cliente entre los grupos de fumadores y no fumadores:

Figura 2.8: Histograma 2-Y – Comparación de costos de seguros entre fumadores y no fumadores



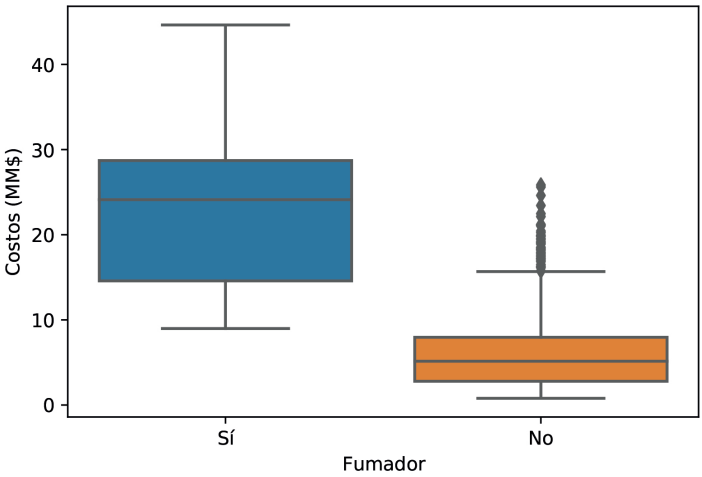
En la figura 2.8 se puede observar que los asegurados no fumadores representan un costo mucho más bajo en comparación con los fumadores. Además, se puede complementar la observación anterior con otra característica del conjunto de datos: se observa que la cantidad de no fumadores es mucho mayor que los fumadores, de manera muy intuitiva.

- **Diagrama *Box plot* o «caja y bigotes»:** este gráfico permite observar la forma en que se distribuyen los valores de determinadas variables, en conjunto con una serie de valores estadísticos de interés para todo analista. Tanto el *box-plot* como el histograma 2-Y permiten relacionar una variable categórica con otra numérica, pero el primero tiene la ventaja que la distribución de la variable numérica puede resumirse en una dimensión, mientras que el histograma 2-Y requiere de dos dimensiones. Esto permite al *box plot* colocar en fila diferentes distribuciones sin traslape y evaluar visualmente las diferencias entre distribuciones de forma mucho más clara.

  Un *box plot* permite obtener dichas estadísticas a través de su representación del cajón o caja (rectángulos rellenos) y los «bigotes» (líneas que se extienden verticalmente desde los rectángulos rellenos). En concreto, los diagramas *box plot* permiten observar la mediana (línea que divide los rectángulos en dos partes), los límites que definen el primer ($Q_1$) y tercer cuartil ($Q_3$), que además representan los bordes del rectángulo, y los *outliers,* que son los valores que se encuentran fuera de los límites que definen los «bigotes».

  En la figura 2.9 se muestran las distribuciones de los costos que pagan las aseguradoras por sus clientes separados en dos agrupaciones, nuevamente: fumadores y no fumadores.

Figura 2.9: Box plot – Comparación de costos de seguros entre fumadores y no fumadores




En la figura anterior, cada rectángulo representa la concentración de un 50% de los valores (rango intercuartílico $Q_{3-1}$) para cada segmento de asegurados. Además, se pueden observar los «bigotes» de ambas distribuciones, los cuales permiten visualizar la variabilidad de cada grupo en términos de los costos que representa para la aseguradora: más específicamente, los bigotes que se encuentran bajo los cajones muestran el intervalo de valores que comprenden el 25% de los valores más bajos de cada distribución primer ($Q_1$). Por otro lado, los bigotes que se encuentran sobre los cajones muestran la variabilidad del 75% de los costos más altos que debe pagar la aseguradora por cada grupo ($Q_3$).

Una observación importante que entrega el *Box plot* es la concentración de los valores de cada agrupación de observaciones: en el caso de la figura 2.9, los no fumadores se encuentran mucho más «concentrados» en valores bajos. Al contrario de lo que ocurre con los fumadores, los cuales poseen costos mucho más altos para la aseguradora y de una manera mucho más variable.

Adicionalmente, la distribución de los fumadores es relativamente asimétrica. Esto se observa en el primer *box plot* al evaluar visualmente las distancias entre la mediana ($Q_2$, la línea horizontal

dentro de la caja) y los bordes de la caja ($Q_1$ y $Q_3$). Se observa que la distancia entre la mediana y el tercer cuartil es más corta que con el primer cuartil, sugiriendo una mayor concentración de fumadores con costos altos mayores a la media.

## 2.2 Limpieza de datos

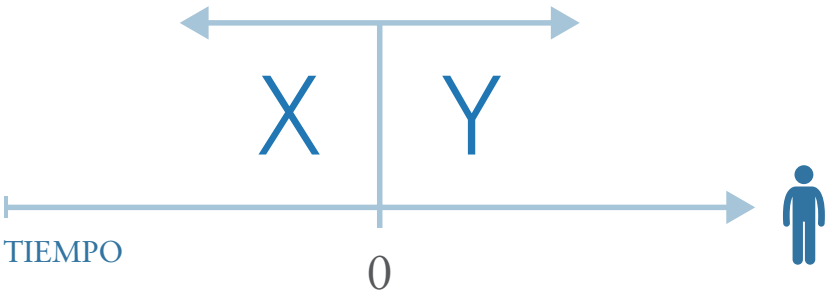
Luego de haber revisado los datos e identificado errores en ellos mediante técnicas para AED, se procede a corregir estos valores perdidos e inconsistencias (VP/I). Contar con un conjunto de datos limpio es clave para la correcta aplicación de modelos predictivos.

Antes de corregir VP/I, debemos preguntarnos: ¿existe un patrón sistemático que genera estos VP/I o son más bien consecuencia del azar? Volviendo al ejemplo de la edad 1970, puede ser que haya un problema de cruce de datos en la etapa de consolidación y tengamos muchos años de nacimientos entremezclados con las edades. En este caso, es razonable utilizar un **reemplazo *ad-hoc,*** es decir, una estrategia diseñada especialmente para hacernos cargo de este patrón. Esta estrategia es calcular la edad a partir de los años de nacimiento y utilizar este valor como reemplazo para la inconsistencia. Este tipo de VP/I se conoce como *not missing at random* (NMAR).

Una pregunta capciosa: ¿cuál es la edad de la persona con un año de nacimiento 1970? Es tentador decir que corresponde a la fecha de hoy menos 1970 (2021-1970=51 años). Esto es un error. Si bien el científico de datos puede encontrarse parado en «hoy» (digamos, 2021, ver figura 2.10) revisando los datos, el registro del cliente puede ser de mucho tiempo atrás. Lo que importa es la definición del «hito 0» de la observación, que en otorgamiento de crédito corresponde al momento en que el cliente fue evaluado en términos de riesgo y finalmente se le otorgó (o no) el crédito. Las variables predictivas solo pueden contener información hasta ese momento, mientras que la variable Y se construye a partir de este punto.

Asumiendo que el cliente fue evaluado en enero de 2002, un ejemplo de Y sería si el cliente tuvo retrasos en sus pagos por al menos tres meses durante el año 2002. La edad del cliente sería 2002-1970=32 años (aproximadamente). Las variables predictivas (matriz X) deben resumir la información hasta enero de 2002. Es tentador utilizar «información del futuro» dentro de la matriz X ya que, como nos encontramos en 2021, podemos ver toda la información del cliente. Por ejemplo, una variable relevante para predecir Y sería la deuda a febrero 2002, sin embargo, esta variable no se puede utilizar ya que está después del hito 0. En otras palabras, ¿cómo podemos preguntarle al cliente, en enero 2002, cuál será su deuda en febrero? Es imposible tener esa información al momento de la evaluación.

Figura 2.10: Línea de tiempo para un ejemplo de otorgamiento de crédito




Veamos otro ejemplo NMAR. Imaginemos que tenemos un conjunto de datos de seguridad social y que, por un error de sistema, cada vez que las personas cumplen 100 años se genera un VP de edad. En este caso, es una mala idea realizar un reemplazo estadístico de estos VP, ya que probablemente los valores generados estarán muy lejos del valor real. Un reemplazo *ad-hoc* sería reemplazar por el máximo valor de la columna, o por 100. Seguramente habrá personas con más de 100 años en la base, pero parece una aproximación bastante razonable.

El ejemplo anterior nos lleva a un mensaje importante de este capítulo: no suele valer la pena realizar un esfuerzo grande por llegar al valor exacto con el cual reemplazar un VP/I. Una buena aproximación basta. Una alternativa a reemplazar por 100 sería realizar todo el cálculo de la edad en base a otros campos, o incluso tratar de preguntarle a la persona su edad. ¿Importa realmente si una persona tiene 100 o 103 años, si lo que buscamos son tendencias generales de la población? Probablemente la respuesta es no.

De forma alternativa, los VP/I pueden ser *missing completely at random* (MCAR), es decir, se generan al azar y no tenemos «pistas» para hacer reemplazos *ad-hoc*. En este caso, lo razonable es realizar algún tipo de reemplazo estadístico con el fin de afectar lo menos posible la distribución de los datos.

Existen varias estrategias de imputación o eliminación de VP/I. Algunas son muy simples y otras más sofisticadas. Si el porcentaje de VP/I es bajo para una columna dada (digamos, menor a 3-5%),

una estrategia simple debería bastar. Dentro de las estrategias simples tenemos la **eliminación de filas** con VP/I o su **reemplazo por una medida de tendencia central adecuada** (promedio, mediana o moda). La primera opción tiene la ventaja de no inventar datos, pero perdemos información potencialmente importante. Si nuestra base de datos no es muy grande y los VP/I están muy dispersos entre las variables, recomendamos la segunda opción.

Si el porcentaje de VP/I es medio (digamos, entre 10 y 30%), ambas opciones no son buenas. Por un lado, la eliminación de filas puede provocar una pérdida de información grave. Por otro lado, reemplazar el 20% de los datos por un valor único puede afectar la distribución de los datos. Lo que sugerimos es generar datos con una cierta dispersión (es decir, no un valor único), respetando la distribución original de los datos. El método recomendado es la **imputación múltiple**, el cual considera todos los VP/I de todas las columnas de forma simultánea, capturando las correlaciones entre variables.

La imputación múltiple es un método bastante sofisticado que no tiene sentido formalizar acá, pero esto no debería asustarnos. Este método se encuentra implementado tanto en R (paquete MICE) como Python (librería *sklearn*), existiendo además en otras plataformas. Al final del día, es un método simple de implementar en un par de líneas de código. La única desventaja es que puede ser bastante lento si la base de datos es muy grande en términos de filas, columnas y porcentaje de VP.

¿Qué pasa si el porcentaje de VP/I es muy alto (digamos, mayor a 50%)? En este caso, la variable podría ser eliminada, ya que ninguna técnica será capaz de rescatar la información faltante, por más sofisticada que sea. Sin embargo, esto es cierto solo cuando enfrentamos valores del tipo MCAR, es decir, sin un patrón detrás que podamos rescatar.

Veamos el siguiente ejemplo. En un proyecto de otorgamiento de crédito para microempresarios nos encontramos con la siguiente variable: «monto declarado de impuestos». Esta variable se refiere al dinero declarado por el solicitante de crédito en el sistema de la

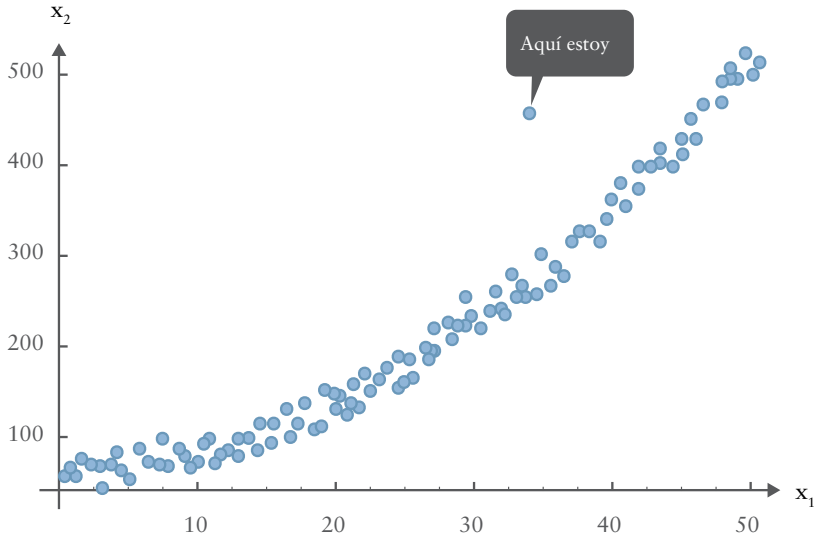
agencia federal que administra temas impositivos. El tema es que esta variable tenía aproximadamente un 50% de VP. Sabíamos que la variable era importante, por eso indagamos un poco más. Resulta que los VP corresponden a postulantes a crédito que no declararon impuestos ya que sus trabajos eran informales. Esto no fue una limitante para que recibieran créditos, pero ciertamente hace el otorgamiento algo más riesgoso. Claramente, los VP son NMAR, ya que hay un patrón que causa la información faltante. ¿La solución? Podemos capturar este patrón con una variable *dummy* o binaria, que toma el valor 1 si el postulante a crédito declara impuestos, o 0 en caso contrario.

Podemos resumir el procedimiento recomendado por nosotros para la corrección de VP/I en los siguientes pasos (notar que esto es en base a experiencias en proyectos más que una ciencia exacta):

- Hacernos cargo, primero, de las inconsistencias. Muchas veces estas no nos entregan pistas para hacer reemplazos *ad-hoc*, por lo que debemos declararlas como VP y luego tratarlas todas juntas.
- Analizar si existen filas que tengan un porcentaje alto de VP, lo recomendable es eliminarlas antes de tratar los VP a nivel de columna.
- Eliminamos las columnas con VP alto y de tipo MCAR.
- Hacerse cargo de las variables nominales, reemplazando por la moda. La imputación múltiple está limitada a variables numéricas en la mayoría de sus implementaciones.
- Hacerse cargo de las variables numéricas, ya sea con reemplazos simples, eliminación de filas o imputación múltiple.

Para finalizar esta sección, también puede ser recomendable eliminar los *outliers* o valores extremos, incluso cuando no son inconsistentes. Esto se debe a que hay modelos predictivos y descriptivos que son sensibles a estos valores. La figura 2.11 presenta una representación gráfica de un *outlier.*

Figura 2.11: Representación de un *outlier* o valor extremo




Los *outliers* pueden ser vistos de forma univariada o multivariada. Una forma univariada de detectar *outliers* es con los gráficos de caja, como el de la figura 2.9. Las observaciones que se escapan de los «bigotes» pueden ser consideradas extremas desde un punto de vista de cuartiles. Otra forma tradicional es descartar observaciones que se encuentran a tres desviaciones estándar de la media. Asumiendo que una variable es normal, el 95% de los datos se encuentra dentro de un intervalo definido por dos desviaciones, mientras que más del 99% está en un intervalo de tres desviaciones.

Para el caso multivariado, los métodos son más complejos y se basan en encontrar zonas densas y menos densas dentro de los datos utilizando *clustering*. Estas técnicas se discuten en el siguiente capítulo de *diagnostic analytics*.

Muchas veces el esfuerzo de detectar *outliers* no se traduce en un mejor desempeño predictivo. Sin embargo, esta tarea puede ser un fin en sí mismo, en el sentido de que permite detectar comportamientos anómalos de clientes. Esto puede llegar a detectar, por ejemplo, algún tipo de patrón fraudulento (Baesens *et al.*, 2015).

## 2.3 Transformación de datos

Después de la etapa de limpieza, nuestra base de datos ya no tiene errores que afecten el desempeño de los modelos. Sin embargo, aún no estamos listos. Por un lado, tenemos variables de fecha, texto u otro que no tienen sentido matemático. Además, es posible aprovechar de mejor manera la información existente generando combinaciones de variables con un potencial mayor.

Cuando hablamos de la transformación de datos, nos referimos a aquellos procesos en donde podemos escalar los valores de las variables, discretizarlas, agregar categorías de variables nominales, codificación *dummy*, entre otras.

La transformación de variables persigue tres objetivos principales:

1. **Dar sentido matemático a las variables:** transformar a variables numéricas todas las fuentes de información disponibles para que un modelo matemático sea capaz de entenderlas.
2. **Mejorar el poder predictivo de las variables:** aprovechar todo el conocimiento y experiencia del *data scientist* con el fin de crear nuevas variables que puedan explicar mejor la variable objetivo. Esto se realiza aplicando operaciones matemáticas sobre las variables originales.
3. **Igualar pesos relativos entre variables:** existen muchos métodos de *diagnostic* y *predictive analytics* que combinan variables y calculan distancias entre ejemplos. Si utilizamos los datos tal y como vienen, compararemos «peras con manzanas», en el sentido que combinaremos variables como edad (en años) e ingreso (en unidades de dinero). Para evitar este problema, necesitamos asegurarnos de que todas las variables tengan la misma escala.

Sobre el primer paso, la transformación principal corresponde a la creación de variables *dummy* o dicotómicas. Volviendo al ejemplo de la variable género, puede ser que los niveles originales sean «M» (masculino) y «F» (femenino). La transformación sugerida es reemplazar los niveles por números: 0 (masculino) y 1 (femenino).

Si una variable tiene más de dos niveles, se crean tantas variables *dummy* como niveles tenga. En los modelos lineales, sin embargo, se suele eliminar una de estas variables antes de entrar a los modelos predictivos, quedando como «categoría de referencia».

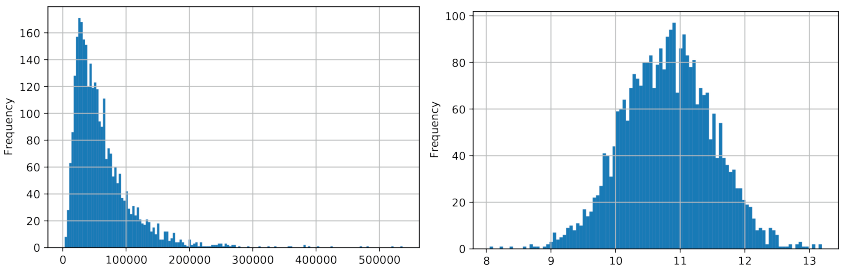
Si las variables son ordinales, hay dos caminos: construir variables *dummy* con cada nivel, sacrificando el orden entre niveles, o bien definir distancias entre niveles y utilizar la variable como numérica. Lo más común es lo primero, sin embargo, existen algunas formas de crear variables ordinales con distancias relativamente claras.

La escala de *Likert*, por ejemplo, define una escala equidistante para una pregunta difícil de cuantificar: «siendo 1 completamente insatisfecho, 3 neutral y 5 completamente satisfecho, ¿cómo se siente en relación el servicio otorgado por nuestra empresa?». Una pregunta como esta «obliga» al participante a definir una escala mental con intervalos equidistantes (la distancia entre «completamente insatisfecho» y «algo insatisfecho» es la misma que entre «algo satisfecho» y «completamente satisfecho»), por esto tiene sentido utilizar variables de escala de *Likert* como numéricas en modelos predictivos.

Continuando con el segundo objetivo, existen diversas formas de mejorar el desempeño predictivo de las variables:

- **Transformación de *Box-Cox* (transformación logarítmica):** la idea es corregir la asimetría a la izquierda de una variable numérica. De esta manera, la variable se acerca a una distribución normal mientras que los valores extremos pierden influencia, ya que no se encuentran tan lejos del resto de los valores. Esta transformación es común en variables con unidades de dinero, como el ejemplo de ingreso en nuestro Laboratorio 1. El efecto de esta transformación se aprecia en la figura 2.12, donde vemos la distribución original (izquierda) con valores altamente concentrados en ingresos bajos, mientras que la variable transformada permite expandir la distribución de dentro hacia afuera. De esa forma es menos sensible a los valores extremos de ingreso.

Figura 2.12: Transformación de Box-Cox (logaritmo natural)



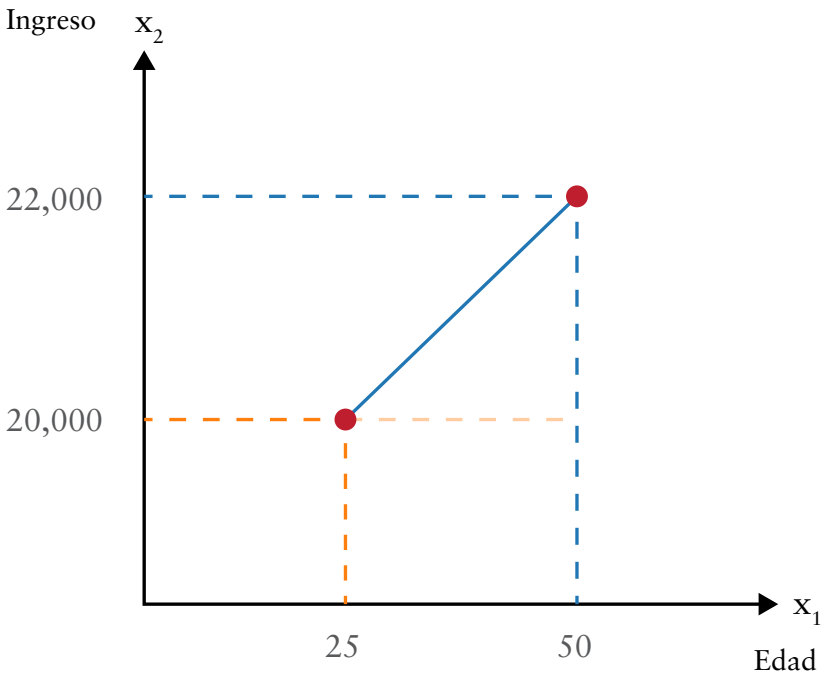

- **Discretización:** esta transformación busca convertir una variable numérica en nominal, sacrificando el orden y las distancias entre categorías. Si bien esto puede llevar a una pérdida de información, puede ser útil en casos puntuales. Por ejemplo, el comportamiento de una variable puede ser no lineal. Algunas veces ocurre esto con la edad: los postulantes a crédito son más riesgosos cuando son jóvenes, ya que tienen una menor estabilidad laboral y responsabilidades. El tramo de edad del medio (digamos, entre 30 y 45 años) suele ser el de mayor estabilidad y el *peak* del ingreso, lo que lo hace el menos riesgoso desde el punto de vista crediticio. Finalmente, el tramo de adulto mayor vuelve a tener un mayor riesgo debido a la opción de enfermedades graves, el mayor costo de vida en general y la dificultad de encontrar empleo en caso de perderlo. Si ocurre este tipo de patrones, una opción que los captura es considerar tres tramos de ingreso, discretizando la variable (Baesens *et al.*, 2016).
  Volvamos al ejemplo de la variable «monto declarado de impuesto» con 50% de VP. Nuestra solución fue crear una variable *dummy* del tipo «declara o no declara impuestos». Sin embargo, sacrificamos información valiosa de los montos declarados. Una solución mejor sería considerar tres categorías: declara poco impuesto, declara mucho impuesto y no declara impuesto. Esto es otro ejemplo de discretización.
  La separación en tramos de edad o monto de impuesto se puede hacer de varias formas: arbitraria, de acuerdo con clasificaciones consideradas en la empresa (menos recomendable pero lo más

simple), percentiles (que el 33.3% de los clientes se encuentren en cada categoría de edad) o buscando umbrales que maximicen la separación entre clases. Esto último es más sofisticado y corresponde a lo que realizan los árboles de decisión que veremos en el capítulo 6.

- **Agregación de categorías:** suele ocurrir que tengamos variables nominales con muchos niveles. En este caso, la creación de variables *dummy* conlleva un gran número de columnas nuevas, introduciendo ruido en los modelos. Otro problema es que muchas categorías se encuentran subrepresentadas en la muestra, llevando a la construcción de variables dicotómicas con un porcentaje extremadamente alto de valores cero. La solución a este problema es combinar niveles afines para reducir el número de *dummies*. Por ejemplo, Chile tiene una división política de 16 regiones. Una empresa podría querer una división territorial más compacta, como, por ejemplo, norte-centro-sur, o simplemente Región Metropolitana versus otras regiones.
- **Construcción de ratios y sumas:** finalmente, podemos diseñar mejores variables combinando atributos numéricos mediante operaciones aritméticas simples. Por ejemplo, se pueden sumar todas las deudas de un cliente y crear la variable «deuda total». Otro caso es el tradicional ratio de endeudamiento, el cual se construye dividiendo la deuda del cliente por el ingreso. Esta variable generada cuenta una mejor historia que las originales, ya que una persona con alta deuda puede tener el ingreso suficiente como para cubrir sus compromisos, a diferencia de una persona que está endeudada muchas veces por sobre su ingreso.

Para finalizar, el tercer objetivo corresponde a igualar los pesos relativos. Pensemos en el siguiente ejemplo: contamos con dos atributos (edad e ingreso) y queremos calcular la distancia entre dos clientes. El primer cliente tiene una edad de 25 años y un ingreso de 20,000 USD/año, mientras que el segundo tiene 50 años y un ingreso de 22,000 USD/año (ver figura 2.13, que no está en escala intencionalmente).

Figura 2.13: Razón por la cual los datos se deben normalizar




Primero entendamos a estos dos clientes: ambos se encuentran cerca del ingreso mínimo en Estados Unidos y la diferencia salarial entre ellos es muy baja. Sin embargo, el primero es joven, de hecho, puede estar recién iniciando su vida laboral, mientras que el segundo ya pasó su *peak* y es probable que su ingreso anual no sufra grandes variaciones. Para todos los efectos, parecen ser clientes muy diferentes.

Veamos ahora la distancia entre ambos clientes. Lo común es utilizar la distancia Euclidiana, que en dos dimensiones es simplemente el teorema de Pitágoras (formalizaremos la distancia Euclidiana en el próximo capítulo). La distancia entre ambos clientes es la raíz cuadrada de $25^2+2000^2$, es decir, 2000.16. En otras palabras, pese a que los clientes son muy parecidos en ingreso y muy diferentes en edad, todo el peso se la lleva la primera variable en el cálculo de la distancia.

¿Por qué? Porque la varianza de una variable como ingreso es mucho mayor que la de la edad, dándole más importancia relativa en un cálculo de distancias. Pero esta relevancia es artificial; no tiene

que ver con la relación entre los predictores y la variable objetivo. Para que ambas variables tengan el mismo peso, existen dos alternativas principales:

- **Escalamiento entre 0 y 1:** llevar las variables a una escala donde el mínimo sea 0 y el máximo sea 1. Para ello, se aplica la siguiente fórmula que depende del mínimo y el máximo de la variable:

Ecuación 2.3

$$x_i^* = \frac{x_i - min(X)}{max(X) - min(X)}$$

- **Estandarización o transformación Z:** llevar las variables a una escala donde la media sea 0 y la desviación estándar sea 1. Para ello, se aplica la siguiente fórmula que depende del promedio y desviación estándar de la variable:

Ecuación 2.4

$$Z_i = \frac{x_i - \bar{X}}{\sigma(X)}$$

El escalamiento tiene la ventaja de que siempre es más intuitivo lidiar con valores positivos, y que el rango de las variables *dummy* es el mismo que las numéricas. Sin embargo, es mucho más sensible a *outliers* e inconsistencias, ya que se basa en el mínimo y el máximo. Por eso es recomendable corregir inconsistencia y aplicar la transformación de *Box-Cox* en variables asimétricas si se va a considerar este primer escalamiento.

Los tres objetivos anteriores se plantean en el orden que recomendamos efectuar las transformaciones: primero asegurarnos de que toda la información sea numérica, luego crear nuevas variables potencialmente relevantes y, finalmente, llevar toda la información

a una escala común. Eso sí, es más fácil realizar las agregaciones primero y después generar las variables *dummy*.

## 2.4 Selección de atributos

Como mencionamos anteriormente, uno de los objetivos de análisis bivariado es la selección de atributos. Este enfoque consiste en seleccionar un subconjunto de variables que sea relevante para el problema que se está abordando, eliminando aquellas que generan ruido y puedan llegar a confundir al modelo. La selección de atributos tiene importantes ventajas (Maldonado *et al.*, 2017):

- Una representación usando menos atributos realza el poder predictivo de los modelos supervisados, disminuyendo su complejidad. Esta simpleza está directamente ligada con el principio de parsimonia o «navaja de Occam», que sugiere que la explicación más sencilla a un fenómeno suele ser la más probable (en igualdad de condiciones). Un modelo con menos variables tendrá menos grados de libertad (flexibilidad) y, por ende, resultará más simple.
- Un modelo con menos variables permite una mejor interpretación del fenómeno, lo que es particularmente importante en analítica de negocios. La comprensión del proceso que generó los datos es clave; por ejemplo, es muy valioso entender los factores que permiten distinguir entre «buenos» y «malos» pagadores de crédito o las razones que causan que los clientes se fuguen.
- Permite una construcción de modelos de forma más rápida, ya que la complejidad computacional de los algoritmos de aprendizaje crece significativamente al recibir más variables como *input*.
- En algunos casos, permite reducir los costos asociados a la adquisición de variables. En riesgo de crédito, por ejemplo, algunos atributos de los clientes pueden ser adquiridos por agencias que venden datos de personas conocidas como *credit bureaus*. Identificando variables externas irrelevantes para el problema evitamos comprarlas en el futuro. Lo mismo ocurre con fuentes

de información internas que pueden requerir de entrevistas largas u otras actividades que tomen tiempo. Valorar el tiempo de ejecutivos y clientes lleva a la estimación de costos indirectos que se pueden evitar.

Volviendo al concepto de la navaja de Occam, otro punto importante es la «maldición de la dimensionalidad» en modelos matemáticos. Este es un problema matemático que ocurre en situaciones de alta dimensionalidad. Consiste en que los datos se vuelven más dispersos entre sí. Esto se debe a que el volumen del espacio aumenta exponencialmente con la cantidad de atributos, lo que provoca que mientras más dimensiones haya, más datos deberíamos tener para construir modelos estables y estadísticamente válidos. En otras palabras, si tenemos muchas columnas y pocas filas en nuestra matriz X, los patrones encontrados serán inestables y muy sensibles al ruido de los datos. Esto trae como consecuencia un pobre desempeño predictivo.

Un concepto muy ligado al de la maldición de la dimensionalidad es el de «sobreajuste». Modelos muy complejos y con muchos grados de libertad se ajustarán demasiado a los datos observados y no lograrán extrapolar adecuadamente los patrones encontrados en los datos al ser utilizados en modo predictivo. Este concepto será discutido en profundidad en las siguientes secciones relacionadas con *predictive analytics*.

### *2.4.1 Taxonomía*

La literatura científica reconoce tres estrategias principales de selección de atributos: (1) métodos de filtro *(filter methods)*, (2) métodos envolventes *(wrapper methods)* y (3) métodos empotrados o embebidos *(embedded methods)*. Adicionalmente, estas estrategias se pueden combinar, resultando en enfoques híbridos (Maldonado *et al.*, 2017).

En primer lugar, los métodos de filtro consideran las propiedades estadísticas de las características para filtrar aquellas que entreguen

menos información. Este tipo de métodos son fundamentales como estrategia de preprocesamiento, para más adelante aplicar algoritmos predictivos.

Los métodos envolventes evalúan diferentes combinaciones de los atributos disponibles utilizando los modelos predictivos, eligiendo aquel subconjunto que trae el mejor desempeño. Estos son computacionalmente más demandantes que los métodos de filtro, ya que requieren la construcción de varios modelos predictivos, pero suelen ofrecer una mayor precisión en los resultados.

Por último, existen los métodos embebidos o empotrados, que eliminan atributos durante la construcción de un único modelo predictivo. Esto se suele realizar introduciendo modificaciones a los modelos predictivos, como por ejemplo la inclusión de funciones de penalización de variables. Existen, además, modelos predictivos que están diseñados para eliminar atributos en su construcción y, por ende, se consideran métodos embebidos. Un ejemplo es el árbol de decisión, que se analizará en el siguiente capítulo. La principal ventaja de los modelos embebidos es que suelen tener una buena capacidad predictiva y son más rápidos que los *wrappers,* ya que solo realizan una evaluación. Sin embargo, son conceptualmente más complejos.

Dado que los métodos envolventes y embebidos están directamente ligados al modelo predictivo, estos serán discutidos en los siguientes capítulos. El resto de esta sección está dedicada a los métodos de filtro.

Los métodos de filtro utilizan diferentes criterios para evaluar los atributos. En este libro se discuten tres:

- **Inutilidad**: cuando una variable nominal está 100%, o casi 100%, concentrada en un único valor, o una variable numérica tiene una desviación estándar cero o muy cercana a cero (digamos, 0.00001 antes de normalizar). Este tipo de variables no aportarán en tareas de aprendizaje (tanto supervisadas como no supervisadas) y son descartables. Esto puede ocurrir con una variable que nadie se preocupó de completar y que se haya terminado de rellenar con ceros.

- **Irrelevancia**: una variable de la matriz X puede ser completamente independiente de la variable objetivo Y. Por ejemplo, la región de residencia puede que no tenga relación con la fuga de los clientes. Existen diferentes técnicas para evaluar la irrelevancia, las cuales se discuten en la siguiente sección.
- **Redundancia**: dos o más variables de la matriz X pueden tener una dispersión adecuada y ser relevantes, sin embargo, pueden ser tan parecidas que no tenga sentido que entren al modelo. Pueden ser incluso la misma variable: muchas veces se recopila información de diferentes fuentes sin notar que puede haber atributos repetidos con diferentes nombres. Para evaluar redundancia se recomienda calcular la matriz correlaciones bivariadas utilizando el rho de Pearson (ecuación 2.2). Lo que buscamos es detectar correlaciones **extremas**: las variables que tienen una correlación de Pearson mayor a 0.9 pueden ser consideradas redundantes.

La redundancia puede traer consecuencias más serias que los otros dos criterios, ya que suele acarrear problemas numéricos en algunos modelos. Por ejemplo, la regresión lineal se basa en la inversión de una matriz, proceso que puede resultar impreciso (o, en el caso extremo, fallar completamente) con la presencia de datos redundantes.

Si detectamos variables redundantes surge la pregunta: ¿cuál de las variables dejamos en nuestra base? Aquí se pueden considerar ciertos criterios de negocios, tales como elegir la que es más confiable (la que tenga menos ruido en el proceso de generación de datos), la más barata de conseguir o la que sea más probable que se encuentre disponible en el futuro. Alternativamente, podemos utilizar un criterio matemático: ante dos o más variables redundantes, elegimos la más relevante.

### *2.4.2 Estrategias de filtro*

Dentro de las diferentes alternativas para filtrar atributos encontramos *tests* de hipótesis y estrategias de ranking. Las primeras estrategias tienen la ventaja de evaluar estadísticamente la relevancia de un atributo, entregando una decisión sobre si el atributo debe ser incluido, o no, en base a un criterio estadístico. Las segundas solo calculan una medida para la contribución de cada atributo, permitiendo ordenar las variables de acuerdo con esta medida. Queda a criterio del analista seleccionar el subconjunto de variables relevantes en base al ranking que entregue la estrategia de filtro.

Un *test* de hipótesis comprende aquellos métodos en que se busca evaluar estadísticamente una determinada afirmación con la finalidad de aceptarla o rechazarla con un determinado nivel de confiabilidad (o probabilidad). Un ejemplo de *test* de hipótesis puede ser la siguiente afirmación: «Es diferente el promedio de salarios del departamento de marketing con respecto al departamento de I+D».

El origen del *test* de hipótesis se encuentra en el área de estudio de la estadística aplicada, donde se suelen situar dos áreas principales: (1) estadística descriptiva, encargada de describir las propiedades de los datos de la muestra y la población; y (2) estadística inferencial, la cual utiliza las propiedades encontradas en la estadística descriptiva para probar hipótesis y obtener conclusiones al respecto.

Los *test* de hipótesis se realizan por medio del cálculo de una medida estadística (de ahora en adelante le llamaremos «estadístico») propia de cada prueba de hipótesis. A continuación, se explica uno de los *test* de hipótesis más conocidos: el *Test* Chi-cuadrado.

#### *2.4.2.1 Test Chi-cuadrado*

El *Test* $\chi^2$ corresponde a una prueba estadística para contrastar si dos variables categóricas son independientes, o no. Este método es bastante útil en clasificación dado que la variable objetivo Y es nominal. Para el caso de las variables en el X que vamos a relacionar con Y, estas deben ser discretizadas previamente en caso de ser numéricas.

El estadístico $\chi^2$ se calcula de la siguiente forma:

Ecuación 2.5

$$\chi_c^2 = \sum_{i=1}^{n} \frac{(O_i - m_i)^2}{m_i}$$

Donde $c$ corresponde a los grados de libertad, $O_i$ el valor observado, $m_i$ el valor esperado, con $i \in \{1, \ldots, n\}$, y $n$ el tamaño de la muestra.

Para explicar la utilidad de esta prueba estadística podemos servirnos del siguiente ejemplo: usted, como analista, posee una base de datos extensa que tiene una gran cantidad de variables con diferentes grados de correlación entre ellas, lo cual puede dificultar la comprensión de los modelos que construya debido a la alta dimensionalidad de sus datos (Lundberg & Lee, 2017) y, a su vez, produce que el entrenamiento de su modelo sea más lento.

Para abordar el problema anterior, se debe identificar la variable objetivo Y y las variables nominales X que desea evaluar, para discriminar si aportarán información a su modelo predictivo. Con dichas variables identificadas, usted puede abordar el problema de la alta dimensionalidad por medio de una técnica de selección de atributos, por ejemplo, el estadístico $\chi_c^2$ que permite evaluar qué variables nominales aportan información nueva a su modelo.

Así, el estadístico $\chi_c^2$ no es algo diferente a un *test* de hipótesis en estadística, donde se busca probar una determinada afirmación, como lo es: «El grado académico de una persona no tiene relación con su sueldo», por ejemplo.

En el caso anterior, la variable «grado académico» sería una variable nominal (X) que buscamos evaluar si posee información relevante para describir, o predecir, el «sueldo» (Y). En este ejemplo, llamaremos «$H_0$» a la hipótesis que deseamos rechazar (hipótesis nula), con la finalidad de comprobar el conjugado; y «$H_1$» a la hipótesis alternativa.

Cabe mencionar que esta prueba estadística se debe utilizar cuando la variable independiente (X) es categórica ya que, por definición, el estadístico $\chi_c^2$ mide la asociación entre dos variables categóricas. Si ambas variables (tanto X como Y) son numéricas, se utiliza la correlación de *Pearson*; y, por último, si X es numérico y existen múltiples clases posibles para Y se puede utilizar la técnica ANOVA (o la prueba *T* si Y es una variable binaria).

Luego de calcular el estadístico $\chi_c^2$, se busca el valor correspondiente dentro de la tabla con los valores teóricos del estadístico $\chi^2$ para los grados de libertad $c$ y el nivel de confiabilidad (se representa con la letra griega $\alpha$, y corresponde al nivel de probabilidad de certeza con que la prueba es considerada).

Ahora bien, el *Test* $\chi^2$ considera dos tipos principales de pruebas: (1) prueba de independencia, que plantea la afirmación «existe una relación entre estas dos variables»; y el (2) *test* de bondad de ajuste, que proporciona una manera de probar qué tan bien una muestra coincide con las características (conocidas o supuestas) de una población más grande, a la cual pretende representar.

Es importante comprender que el objetivo principal del estadístico $\chi^2$ es medir la diferencia entre frecuencias observadas y esperadas de los resultados acerca de una determinada variable, que depende del tamaño de la diferencia entre los valores reales y observados $(O_i - m_i)$, los grados de libertad $(c)$ y el tamaño de la muestra $(n)$.

Ambas aplicaciones del Test $\chi^2$ son de gran utilidad en el área de *Data Analytics*, ya que a través de (1) se puede identificar si una determinada variable es independiente de otra (por lo cual no aportan la misma información al modelo), y, por otro lado, mediante (2) se puede concluir si una muestra de datos se ajusta a las características de una población, es decir, si la muestra es representativa de esa población o si se ajusta a una determinada distribución estadística.

### *2.4.3 Fisher Score*

El algoritmo de puntuación de Fisher, o método de *Fisher Score*, corresponde a otro de los métodos de selección de variables más

utilizados en la literatura, donde se calcula la importancia de cada variable en base a la diferencia entre la media de dos distribuciones. A diferencia del *test* Chi-cuadrado, esta medida relaciona una variable numérica (dentro de la matriz X) con la variable objetivo Y (binaria) en un problema de clasificación.

El cálculo del *Fisher score* $F(j)$ para una variable explicativa j se realiza de la siguiente forma:

Ecuación 2.6

$$F(j) = \left| \frac{\mu_j^+ - \mu_j^-}{\left(\sigma_j^+\right)^2 - \left(\sigma_j^-\right)^2} \right|$$

El parámetro $\mu_j^+ \ (\mu_j^-)$ corresponde a la media para la característica *j* en la clase positiva (negativa), y $\sigma_j^+ \ (\sigma_j^-)$ es su respectiva desviación estándar. La definición de clase positiva o negativa viene dada por Y. Una vez que se obtienen todos los *F (j)*, estos se ordenan de forma decreciente. Aquellos *F (j)* más cercanos a 0 (esta medida es siempre positiva) se consideran irrelevantes.

## Referencias


Baesens, B. (2014). *Analytics in a* Big Data *World*. New Jersey: John Wiley and Sons.

Baesens, B., Van Vlasselaer, V. & Verbeke, W. (2015). *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*. New Jersey: John Wiley and Sons.

Baesens, B., Roesch, D. & Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. New Jersey: John Wiley and Sons.

Han, J., Kamber, M. & Pei, J. (2011). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Maldonado, S., Bravo, C., Pérez, J. & López, J. (2017). «Integrated framework for profit-based feature selection and SVM classification in credit scoring». *Decision Support Systems,* 104, 113-121.

Thomas, L. C. (2009). *Consumer Credit Models: Pricing, Profit and Portfolios*. New Jersey: Wiley.

# Laboratorio 1: Preprocesamiento

Los datos, en el mundo real, suelen estar incompletos y tener inconsistencias. Por esto es necesario prepararlos antes de ejecutar modelos de analítica. En este primer laboratorio prepararemos datos de riesgo de crédito para que, en los próximos laboratorios, encontremos patrones valiosos para la toma de decisiones.

Para partir, descargamos las librerías a utilizar. Esto es necesario hacerlo la primera vez que queremos ejecutar estos códigos. Tenemos bibliotecas como *numpy*, que se utiliza para hacer operaciones matemáticas, o *scikit-learn*, diseñada para el aprendizaje de máquinas.

```
!pip -q install numpy gensim scikit-learn
```

Luego es necesario importar las librerías con las que vamos a trabajar. Lo haremos cada vez que queramos correr este código. Aquí se muestran solo algunas librerías (el detalle completo está en el sitio web del libro).

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
```

## 1. Subir un archivo de Google Drive

El siguiente paso es montar Google Drive para poder enlazar nuestra base de datos. De esta forma podremos cargar después nuestra base de datos creditSIM.xlsx (que debemos tener en una carpeta «*Colab Notebooks*» en nuestro Drive claramente identificada).

```
from google.colab import drive drive.mount('/content/gdrive')
```

Una vez cargado nuestro Drive, llamamos a nuestra base de datos:

```
ruta_data_set = 'gdrive/My Drive/creditSIM.xlsx'
data_set = pd.read_excel(ruta_data_set)
```

Descripción de variables de la base de datos:

1. ID: Número de cliente.
2. AgnosDirec: Años que el cliente lleva viviendo en la misma dirección.
3. AgnosEmpleo: Años de antigüedad del cliente en el empleo.
4. DeudaExt: Deuda del cliente externa al banco.
5. DeudaInt: Deuda del cliente en el banco. Por ejemplo, otros créditos.
6. Edad: Edad del cliente.
7. Ingreso/Ingreso2: Ingreso del cliente (recopilados de dos fuentes diferentes).
8. Nacionalidad: Nacionalidad del cliente.
9. NivelEdu: Nivel educacional del cliente.
10. VarObj: Variable objetivo. Si el cliente es un mal pagador («cae en bancarrota») en caso de recibir el crédito. Si el cliente cae en bancarrota toma el valor de S y, en el caso contrario, de N.

## 2. Análisis exploratorio de datos (AED)

El AED consiste en examinar los datos para entenderlos, descubrir relaciones entre variables y encontrar posibles inconsistencias. Para esto, se analiza estadística descriptiva y gráficos.

### *2.1 Estadística descriptiva*

Para analizar el comportamiento de los datos de las variables numéricas, obtenemos la media, desviación estándar, mínimo, máximo y cuartiles. Mientras que para las variables categóricas obtenemos

la cantidad de categorías de la variable *(unique)*, la moda *(top)* y la frecuencia de la moda *(freq)*.

```
data_set.describe(include='all').round(1)
```

Tabla 1: Descripción del data_set

| | ID | AgnosDirec | AgnosEmpleo | DeudaExt | DeudaInt | Edad | Ingreso | Ingreso2 | Nacionalidad | NivelEdu | VarObj |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 3000.0 | 2999.0 | 2998.0 | 2999.0 | 2996.0 | 2998.0 | 2998.0 | 3000.0 | 3000 | 3000 | 3000 |
| **unique** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1 | 5 | 2 |
| **top** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | N | SupCom | N |
| **freq** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 3000 | 979 | 2363 |
| **mean** | 101500.5 | 7.1 | 7.1 | 4531.0 | 2275.7 | 37.7 | 62567.7 | 52152.7 | NaN | NaN | NaN |
| **std** | 866.2 | 7.0 | 6.7 | 7084.8 | 5409.8 | 36.6 | 49385.7 | 42246.0 | NaN | NaN | NaN |
| **min** | 100001.0 | 0.0 | 0.0 | 21.6 | 8.3 | 18.0 | -13123.6 | -9723.7 | NaN | NaN | NaN |
| **25%** | 100750.8 | 2.0 | 2.0 | 1110.2 | 416.0 | 29.0 | 30282.7 | 25109.6 | NaN | NaN | NaN |
| **50%** | 101500.5 | 5.0 | 5.0 | 2278.7 | 958.3 | 36.5 | 49516.1 | 40632.3 | NaN | NaN | NaN |
| **75%** | 102250.2 | 10.0 | 10.0 | 5093.8 | 2318.3 | 44.0 | 78851.5 | 66206.2 | NaN | NaN | NaN |
| **max** | 103000.0 | 61.0 | 47.0 | 123939.2 | 190648.9 | 1970.0 | 536488.4 | 513618.6 | NaN | NaN | NaN |

De la tabla anterior, podemos identificar VP (gracias a la fila *count*) e inconsistencias (gracias a las filas *min* y *max*), tales como una edad de 1970 o ingresos negativos.

También podemos observar cómo vienen los datos de cada variable desplegando en pantalla una muestra de ellos:

```
print(data_set)
```

| | ID | AgnosDirec | AgnosEmpleo | ... | Nacionalidad | NivelEdu | VarObj |
|---|---|---|---|---|---|---|---|
| 0 | 100001 | 4.0 | 2.0 | ... | N | SupInc | S |
| 1 | 100002 | 28.0 | 25.0 | ... | N | Med | N |
| 2 | 100003 | 2.0 | 3.0 | ... | N | SupCom | N |
| 3 | 100004 | 11.0 | 14.0 | ... | N | Bas | N |
| 4 | 100005 | 3.0 | 1.0 | ... | N | SupInc | S |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 2995 | 102996 | 7.0 | 6.0 | ... | N | SupInc | N |
| 2996 | 102997 | 3.0 | 3.0 | ... | N | Med | S |
| 2997 | 102998 | 3.0 | 5.0 | ... | N | SupInc | S |
| 2998 | 102999 | 13.0 | 8.0 | ... | N | Med | S |
| 2999 | 103000 | 2.0 | 6.0 | ... | N | SupInc | N |

Una forma más cómoda de visualizar valores perdidos es la siguiente:

```
data_set.isnull().sum()
```

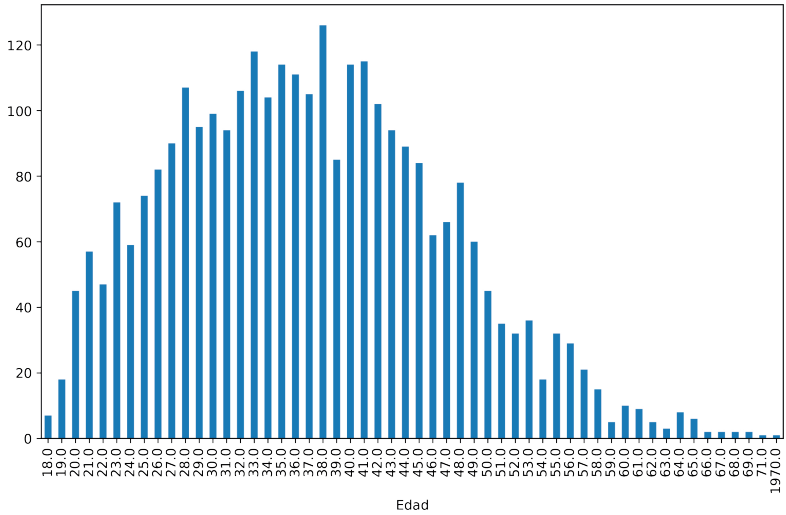
| | |
|---|---|
| ID | 0 |
| AgnosDirec | 1 |
| AgnosEmpleo | 2 |
| DeudaExt | 1 |
| DeudaInt | 4 |
| Edad | 2 |
| Ingreso | 2 |
| Ingreso2 | 0 |
| Nacionalidad | 0 |
| NivelEdu | 0 |
| VarObj | 0 |

## *2.2 Gráficos*

Los gráficos nos permiten identificar patrones univariados y bivariados, también visualizar datos y valores atípicos. Veamos primero el *gráfico de frecuencias* estándar para la variable edad:

```
data_set['Edad'].value_counts().sort_index().plot.bar(figsize=(10,6))
plt.xlabel('Edad')
plt.show()
```

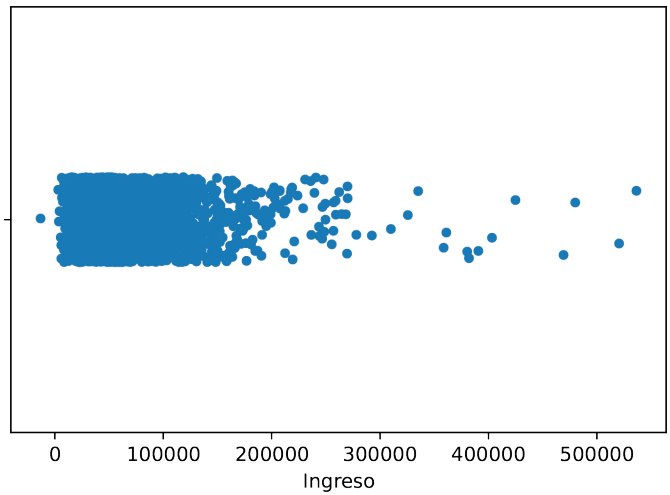
Figura 1: Gráfico de frecuencias estándar de la variable Edad




Vemos que existe una persona con edad 1970, lo que no es posible, por lo tanto, tenemos que corregir esta inconsistencia. El histograma de puntos permite visualizar valores atípicos *(outliers)*, por ejemplo, en la variable ingreso:

```
sn.stripplot(x='Ingreso', data=data_set)
```

Figura 2: Histograma de puntos de la variable Ingreso



Todo lo anterior se asocia al **análisis univariado**, es decir, solo una variable a la vez. Podemos realizar **análisis bivariado** con el *histograma color*, y así ver la relevancia de un atributo con relación a la variable objetivo. En este caso analizamos las variables de años en el mismo empleo y nivel educacional:

```
data_set.groupby(['AgnosEmpleo','VarObj']).size().unstack().plot.bar(stacked=True)
plt.xscale('linear')
data_set.groupby(['NivelEdu', 'VarObj']).size().unstack().plot.bar(stacked=True)
```

Figura 3: Hist. de color de la variable AgnosEmpleo

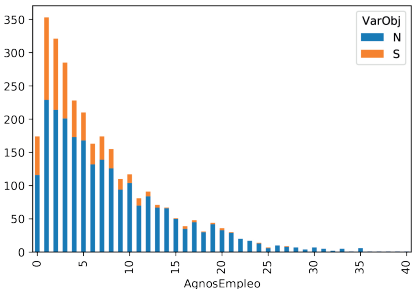
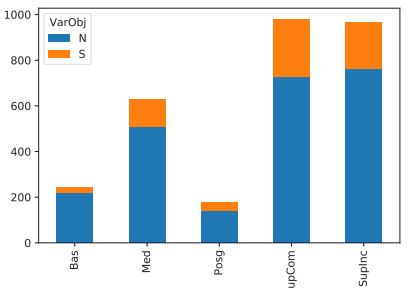



Figura 4: Hist. de color de la variable NivelEdu




En el primer gráfico vemos claramente que la variable (1) es relevante y (2) se comporta de forma esperada: a mayor años trabajando en el mismo empleo (nos movemos a la derecha por el eje X), los clientes son, en promedio, menos riesgosos (la proporción de azul en las barras va creciendo).

El segundo gráfico es menos claro en el patrón, pero da la impresión de que el riesgo es mayor en categorías como educación superior completa (SupCom) e incompleta (SupInc) con relación a categorías con menor nivel educacional, como educación básica. Este patrón es claramente inconsistente con lo que uno esperaría de la realidad, que es «a mayor nivel educacional, menor riesgo».

¿Qué podría estar causando esta inconsistencia? En riesgo de crédito existe el problema del **sesgo de selección**: nuestros datos consisten en solo los clientes que fueron aceptados por la entidad financiera. Probablemente, muchos clientes con un nivel educacional bajo fueron rechazados, quedando aquellos que mostraban un alto potencial crediticio (ingreso alto, bajas deudas, etcétera). Por

otro lado, los clientes con estudios universitarios pueden haber sido aceptados sin solicitar mayores antecedentes, lo que puede explicar su mayor riesgo intrínseco.

Una variable inconsistente con la realidad debería ser descartada con fines predictivos, ya que introduce un sesgo indeseado en el modelo. Si lo que buscamos es reemplazar el modelo original que acepta y rechaza clientes, tenderíamos a favorecer a potenciales clientes con un menor nivel educacional. Este análisis es muy importante ya que la máquina no puede hacerlo (no sabe lo que se espera de una variable).

Cuando los patrones no son muy claros, podemos hacer el histograma color con proporciones en vez de conteos:

```
data_set.groupby(['AgnosEmpleo', 'VarObj']).size().groupby(level=0).
apply(lambda x: 100 * x / x.sum()).unstack().plot.bar(stacked=True)
plt.xscale('linear')
data_set.groupby(['NivelEdu', 'VarObj']).size().groupby(level=0).apply(lambda
x: 100 * x / x.sum()).unstack().plot.bar(stacked=True)
```

Figura 5: Hist. de color de la variable AgnosEmpleo

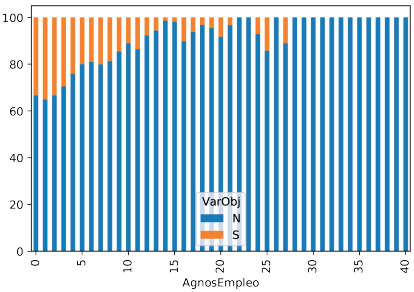
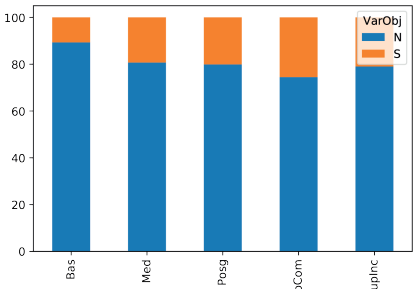



Figura 6: Hist. de color de la variable NivelEdu




El gráfico de cajas o *box plot* es bastante útil como alternativa al histograma color, ya que puede entregar la misma información de la distribución, pero en una sola dimensión (vertical). Esto permite presentar las diferentes distribuciones de forma separada. Así es más fácil contrastar las diferencias:

```
sn.boxplot(x=»VarObj», y=»AgnosEmpleo», data=data_set)
```

Finalmente, el gráfico de dispersión *(scatter plot)* permite ver redundancia y relaciones bivariadas entre dos variables numéricas:

```
data_set.plot(kind='scatter', x='AgnosDirec', y='AgnosEmpleo', grid=True)
```

Figura 7: Box plot de VarObj y AgnosEmpleo

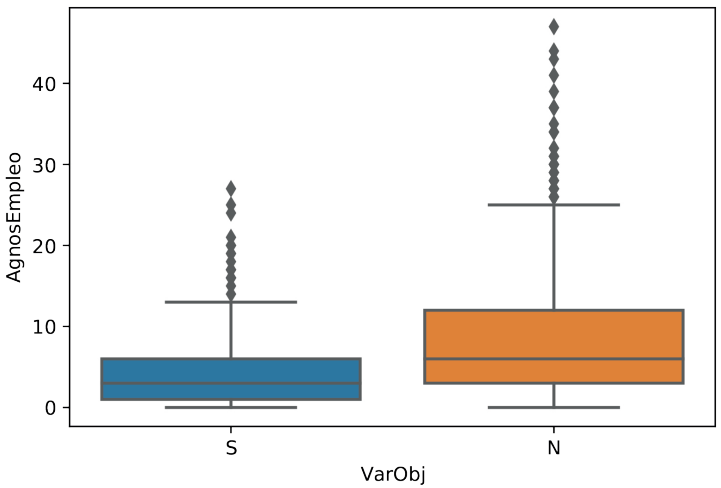
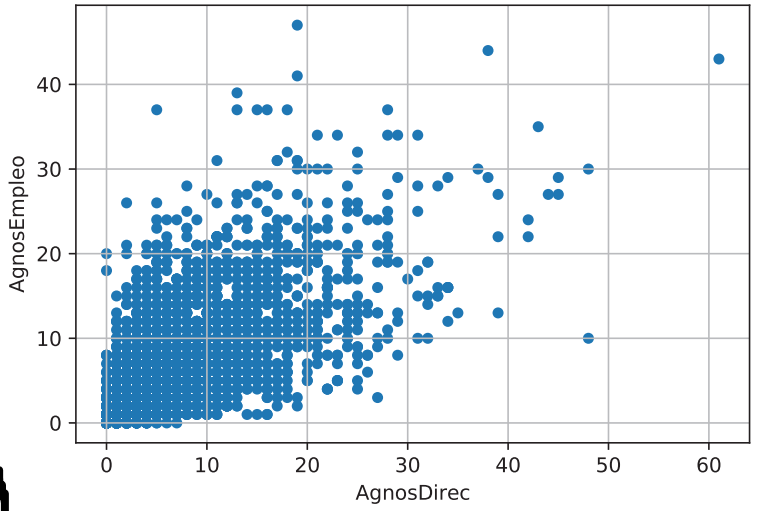



Figura 8: Scatter plot de AgnosDirec y AgnosEmpleo



## 3. Limpieza de datos

La limpieza de datos consiste en eliminar las incongruencias encontradas en el análisis exploratorio de datos, como valores perdidos y valores atípicos, para poder trabajar la base de datos sin que estos alteren los resultados.

Primero declaramos inconsistencias como valores perdidos, para luego imputarlos.

### *3.1 Inconsistencias*

La variable Edad no puede tener valores mayores a 100, dejando los mayores a este número como valores nulos. Lo mismo se realiza para ingresos menores a 0.

```
data_set.loc[data_set['Ingreso'] < 1, ['Ingreso']] = np.nan
data_set.loc[data_set['Ingreso2'] < 1, ['Ingreso2']] = np.nan
data_set.loc[data_set['Edad'] > 100, ['Edad']] = np.nan
```

### *3.2 Imputar valores perdidos*

Para hacernos cargo de los valores perdidos, proponemos tres alternativas (solo la primera se implementa, las restantes se presentan por completitud).

*Alternativa 1: Reemplazo por mediana / moda*

Esta alternativa la utilizamos cuando los valores perdidos son menores al 5% del total de los datos. Reemplazamos por la mediana en las variables numéricas (la media también se puede utilizar):

```
data_set['Edad'].fillna(data_set['Edad'].median(skipna =True),inplace=True)
data_set['AgnosDirec'].fillna(data_set['AgnosDirec'].median(skipna =True),
inplace=True)
data_set['AgnosEmpleo'].fillna(data_set['AgnosEmpleo'].median(skipna =True),
inplace=True)
data_set['Ingreso'].fillna(data_set['Ingreso'].median(skipna =True), inplace=True)
```

```
data_set['Ingreso2'].fillna(data_set['Ingreso2'].median(skipna =True),
inplace=True)
data_set['DeudaExt'].fillna(data_set['DeudaExt'].median(skipna =True),
inplace=True)
data_set['DeudaInt'].fillna(data_set['DeudaInt'].median(skipna =True),
inplace=True)
```

Reemplazamos por la moda en las variables categóricas (esto es solo por completitud ya que «NivelEdu» será descartada):

```
data_set['NivelEdu'].mode()
data_set['NivelEdu'].fillna(mode[0], inplace=True)
```

*Alternativa 2: Eliminación de casos (filas)*

Esta alternativa también la podemos utilizar cuando los valores perdidos son menores al 5% del total de los datos y tenemos una gran cantidad de registros. La ventaja es que no inventamos valores como en el caso anterior, pero perdemos la información de las otras variables en la fila. La ventaja es que se puede implementar de forma muy simple:

```
data_set = data_set.dropna(axis=0, how='any')
```

*Alternativa 3: Imputación múltiple*

La imputación múltiple consiste en rellenar todos los valores faltantes de forma simultánea utilizando simulación. La idea es respetar tanto las distribuciones de cada variable como las correlaciones entre ellas. Es lo más sofisticado que sugerimos y es altamente recomendable cuando tenemos una base de datos pequeña o mediana, junto a un porcentaje más alto de valores perdidos (digamos, 5 a 30%).

Nota: este método no suele funcionar para variables categóricas (hay que reemplazarlas por la moda antes).

```
imp = IterativeImputer(max_iter=10, random_state=12700)
num = data_set.select_dtypes(include=[«number»])
cat = data_set.select_dtypes(exclude=[«number»])
names = num.columns
```

```
num = pd.DataFrame(imp.fit_transform(num))
num.columns = names
data_set = pd.concat([num, cat], axis=1)
```

Corroboramos que no existan valores perdidos en la base de datos:

```
data_set.isnull().sum()
```

| | |
|---|---|
| ID | 0 |
| AgnosDirec | 0 |
| AgnosEmpleo | 0 |
| DeudaExt | 0 |
| DeudaInt | 0 |
| Edad | 0 |
| Ingreso | 0 |
| Ingreso2 | 0 |
| Nacionalidad | 0 |
| NivelEdu | 0 |
| VarObj | 0 |

Antes de continuar, eliminamos la variable ID ya que no aporta información al modelo (es un correlativo):

```
del data_set['ID']
```

## 4. Transformación

La transformación de datos tiene tres objetivos principales: (1) mejorar la capacidad predictiva de una variable, (2) dar sentido matemático a las variables y (3) igualar pesos relativos. Este tercer objetivo lo dejaremos para el cierre del laboratorio.

### *4.1 Agregación*

Si una variable es categórica, podemos agrupar alternativas. Para «NivelEdu», por ejemplo, podemos construir «SupCom» + «Posg» = «Completa». Esto permite reducir la cantidad de variables binarias cuando le damos sentido matemático. De todos modos, esta variable será descartada; solo la presentamos como un ejemplo de código.

```
data_set['NivelEdu'] = data_set['NivelEdu'].replace('SupCom', 'Completa')
data_set['NivelEdu'] = data_set['NivelEdu'].replace('Posg', 'Completa')
```

Si una variable es numérica, la agregación puede ser vista como la construcción de nuevas variables que resultan de la combinación de las originales. Por ejemplo, podemos sumar la deuda interna y externa del cliente, lo que nos da la deuda total. Con esto calculamos el ratio deuda versus ingreso, que representa el endeudamiento real del cliente:

DeudaIng = (DeudaInt + DeudaExt) /Ingreso

```
data_set['DeudaIng'] = (data_set['DeudaInt'] + data_set['DeudaExt'])/data_set['Ingreso']
```

## *4.2 Transformación de Box-Cox (corrección de asimetrías)*

Si las variables son asimétricas a la izquierda (la media está muy cerca de los valores más pequeños), podemos aplicar la función logaritmo para expandir la distribución de dentro hacia afuera. Primero veamos la distribución de variables asimétricas:

```
data_set['Ingreso'].plot(kind='hist',bins=150, grid=True )
data_set['Ingreso2'].plot(kind='hist',bins=150, grid=True )
data_set['DeudaExt'].plot(kind='hist',bins=60, grid=True)
data_set['DeudaInt'].plot(kind='hist',bins=57, grid=True )
```

Figura 9: Hist. de la variable Ingreso

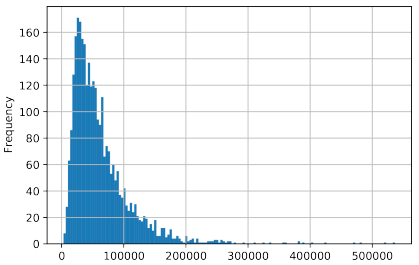



Figura 10: Hist. de la variable Ingreso2

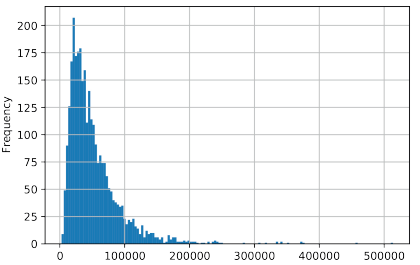


Figura 11: Hist. de la variable DeudaExt

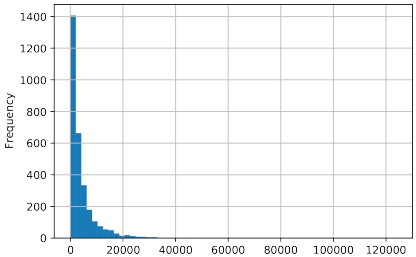
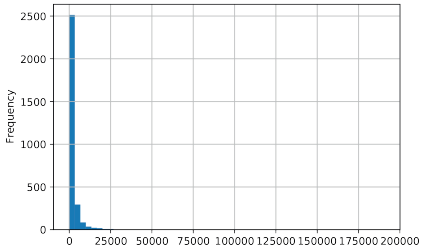



Figura 12: Hist. de la variable DeudaInt




Como vemos, las variables son efectivamente asimétricas, por lo que le aplicamos su logaritmo (sumar 1 evita que se indefina la función en caso de haber ceros):

```
data_set['Ln_Ingreso'] = np.log(data_set['Ingreso']+1)
data_set['Ln_Ingreso2'] = np.log(data_set['Ingreso2']+1)
data_set['Ln_DeudaExt'] = np.log(data_set['DeudaExt']+1)
data_set['Ln_DeudaInt'] = np.log(data_set['DeudaInt']+1)
```

Ahora grafiquemos estas nuevas variables:

```
data_set['Ln_Ingreso'].plot(kind='hist',bins=100, grid=True )
data_set['Ln_Ingreso2'].plot(kind='hist',bins=100, grid=True )
data_set['Ln_DeudaExt'].plot(kind='hist',bins=40, grid=True )
data_set['Ln_DeudaInt'].plot(kind='hist',bins=40, grid=True )
```

Figura 13: Hist. de la variable Ln_Ingreso

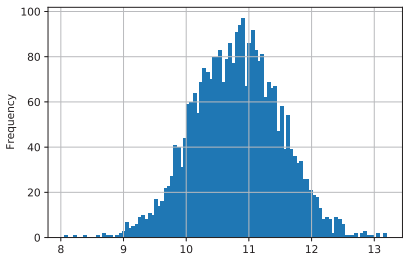



Figura 14: Hist. de la variable Ln_Ingreso2

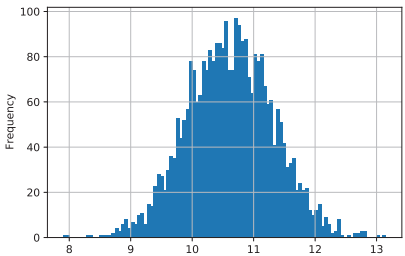


Figura 15: Hist. de la variable Ln_DeudaExt

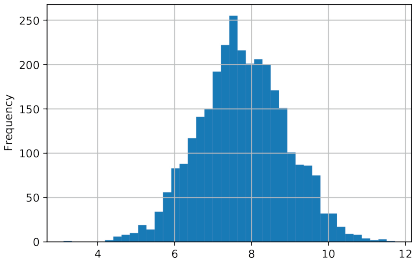
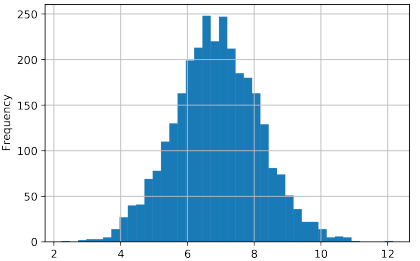



Figura 16: Hist. de la variable Ln_DeudaInt




Como vemos en los gráficos anteriores, la asimetría se corrige y las distribuciones nuevas se acercan a una normal. Eliminamos las variables «Ingreso», «DeudaExt» y «DeudaInt» para quedarnos solo con sus logaritmos. Aprovechamos de eliminar «NivelEdu» por ser una variable sesgada.

```
del data_set['Ingreso']
del data_set['Ingreso2']
del data_set['DeudaExt']
del data_set['DeudaInt']
del data_set['NivelEdu']
```

*Dar sentido matemático a variables nominales*

Transformamos las variables categóricas de la base de datos a variables binarias:

```
data_set = pd.get_dummies(data_set, columns = [«VarObj»], drop_first = True)
data_set = pd.get_dummies(data_set, columns = [«Nacionalidad»])
```

## 5. Filtros (selección de atributos)

Los filtros se utilizan para elegir un subconjunto de variables relevantes, eliminando las que generan ruido antes de correr los modelos. Veremos tres criterios:

## *5.1 Inutilidad*

Este criterio consiste en descartar de la base de datos las variables muy concentradas. Para esto calculamos la desviación estándar de las variables:

```
data_set.std()
```

| | |
|---|---|
| AgnosDirec | 6.960621 |
| AgnosEmpleo | 6.650822 |
| Edad | 9.783355 |
| DeudaIng | 0.084710 |
| Ln_Ingreso | 0.693170 |
| Ln_Ingreso2 | 0.701547 |
| Ln_DeudaExt | 1.140973 |
| Ln_DeudaInt | 1.277791 |
| VarObj_S | 0.409028 |
| Nacionalidad_N | 0.000000 |

Descartamos «Nacionalidad» porque su desviación estándar es 0, es decir, la variable siempre toma el mismo valor.

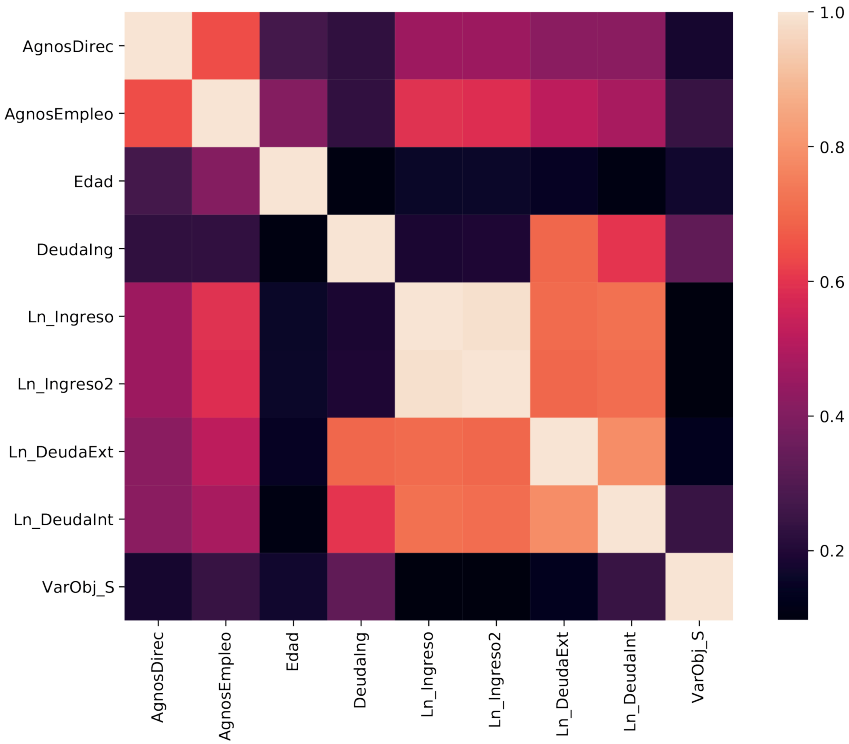
```
del data_set['Nacionalidad_N']
```

## *5.2 Redundancia*

Este criterio consiste en encontrar las variables que están muy correlacionadas entre sí, dejando solo una, puesto que variables redundantes no aportan información nueva. Lo primero es visualizar la matriz de correlaciones bivariadas mediante un *heatmap* o mapa de calor:

```
corrmat = data_set.corr().abs()
f, ax = plt.subplots(figsize=(10, 7))
sn.heatmap(corrmat, square=True);
```

Figura 17: Matriz de correlaciones bivariadas




En la figura anterior vemos que la variable «Ingreso» e «Ingreso2» están altamente correlacionadas (0.986, como se observa en el código de abajo), por lo que tenemos que eliminar una. Podemos analizar la relevancia de ambas para saber cuál elegir.

```
corrmat.loc[«Ln_Ingreso»]
```

```
..
Ln_Ingreso2        0.986428
...
```

## *5.3 Relevancia*

Finalmente, este criterio consiste en ver qué variables de la matriz X están más correlacionadas con la variable objetivo Y. Utilizaremos

el *test* chi cuadrado, que calcula la relación entre dos variables nominales. Las variables numéricas las podemos discretizar.

Calcularemos la relevancia entre las variables redundantes «Ingreso» e «Ingreso2», las cuales discretizaremos en 5 *bins* o intervalos:

```
cut_points = data_set['Ln_Ingreso'].quantile([0, .2, .4, .6, .8, 1])
Ingreso_binned = pd.cut(data_set['Ln_Ingreso'], bins =cut_points, include_
lowest= True)
cut_points = data_set['Ln_Ingreso2'].quantile([0, .2, .4, .6, .8, 1])
Ingreso2_binned = pd.cut(data_set['Ln_Ingreso2'], bins =cut_points, include_
lowest= True)
```

Ahora que tenemos las variables discretizadas «Ingreso_binned» e «Ingreso2_binned», procedemos a calcular el *test* chi cuadrado:

```
tbl = pd.crosstab(index= Ingreso_binned,columns=data_set['VarObj_S']) stat, p,
dof, expected = chi2_contingency(tbl)
print('Valor chi cuadrado = %d' % stat)
print('P-valor =%f' % p)
tbl = pd.crosstab(index=Ingreso2_binned, columns=data_set['VarObj_S']) stat, p,
dof, expected = chi2_contingency(tbl)
print('Valor chi cuadrado = %d' % stat)
print('P-valor =%f' % p)
```

```
Valor chi cuadrado = 37
P-valor =0.000000
Valor chi cuadrado = 38
P-valor =0.000000
```

Ambas tienen casi igual relevancia, lo que es esperado dada su alta correlación. Sin embargo, Ln_Ingreso2 es ligeramente más relevante (Valor chi cuadrado = 38 versus Valor chi cuadrado = 37), por lo que eliminaremos la variable Ingreso.

```
del data_set['Ln_Ingreso']
```

*Selección de variables más relevantes*

Seleccionaremos las variables más relevantes con el método *SelectKbest*. Primero definimos como X las variables independientes y como Y la variable dependiente:

```
X = data_set.loc[:, data_set.columns != 'VarObj_S']
y = data_set.loc[:, data_set.columns == 'VarObj_S']
X.shape #esto nos dice que tenemos 7 variables
```

```
(3000, 7)
```

Ahora aplicamos la función *SelectKBest* con el *test* F de Fisher (opción por defecto, similar a ANOVA) para determinar las variables más relevantes. Alternativamente, se puede utilizar el *test* chi cuadrado («chi2») si las variables son principalmente nominales:

```
X_new = SelectKBest(k=5).fit(X, y.values.ravel())
#Veamos test F y p-valor de todas las variables (no importa que diga k=5; es un número arbitrario)
print(X_new.scores_)
print(X_new.pvalues_)
#Vemos que todas las variables son relevantes para el problema (p-valor<0.05). No descartamos variables
```

```
[100.59100772 185.82963936 96.67536674 367.43291817 28.06912326
55.02637511 191.78625866]
[0. 0. 0. 0. 0.00000013 0.
 0. ]
```

## *Normalización*

El último paso del preprocesamiento será escalar los datos entre 0 y 1 para llevarlos a una escala común. Esto no es necesario para todos los métodos de clasificación, como veremos en los laboratorios siguientes.

```
VarObj = data_set['VarObj_S']
scaler = MinMaxScaler()
scaler_data = scaler.fit(data_set)
data_set_scaled = pd.DataFrame(scaler_data.transform(data_set), index=data_set.index, data_set_scaled['VarObj_S'] = VarObj
print(data_set_scaled)
```

```
AgnosDirec AgnosEmpleo Edad ... Ln_DeudaExt Ln_DeudaInt VarObj_S
0 0.065574 0.042553 0.415094 ... 0.702512 0.676233 1
1 0.459016 0.531915 0.584906 ... 0.728745 0.665300 0
2 0.032787 0.063830 0.377358 ... 0.443684 0.478572 0
3 0.180328 0.297872 0.471698 ... 0.718507 0.558412 0
4 0.049180 0.021277 0.283019 ... 0.420603 0.484673 1
... ... ... ... ... ... ... ...
2995 0.114754 0.127660 0.188679 ... 0.602101 0.500450 0
2996 0.049180 0.063830 0.320755 ... 0.746504 0.537673 1
2997 0.049180 0.106383 0.207547 ... 0.729762 0.604681 1
2998 0.213115 0.170213 0.603774 ... 0.607499 0.653348 1
2999 0.032787 0.127660 0.113208 ... 0.455049 0.403166 0
[3000 rows x 8 columns]
```

## 6. Descargar base de datos preprocesada

Descargamos la base de datos preprocesada en Google Drive. Con esto el archivo quedará guardado en My Drive (Mi Unidad, si Drive está configurado en español). Guardaremos la base sin escalar, ya que la escala original será necesaria después.

```
data_set.to_excel('creditSIM_preprocess.xlsx')
!cp creditSIM_preprocess.xlsx «gdrive/My Drive/»
```

# Aplicación 1: Segmentación de clientes y *Targeting*

La interacción con el cliente en los últimos años se ha convertido en parte fundamental de la dinámica organizacional, posicionándolo en el centro de las estrategias desarrolladas. Esto debido a su impacto directo en conceptos como ingresos, costos, riesgos, productividad y la propia experiencia ofrecida. En paralelo, se han incorporado diferentes conceptos que permiten integrar el comportamiento del cliente en la dinámica organizacional, mencionando, por ejemplo, la visión 360° del cliente (visión unificada e íntegra de la experiencia del cliente), orientación al servicio y la experiencia, ventas cruzadas *(cross-selling)* y venta adicional *(up-selling)*.

Una parte importante de esto es comprender al cliente en su totalidad, conociendo sus gustos, necesidades, hábitos de compra, características demográficas y socioeconómicas. Para esto, generar perfiles de clientes es una actividad fundamental. Segmentar permite focalizar esfuerzos y optimizar los recursos en base a las características de los clientes.

## 1. Segmentación utilizando análisis RFM

*Profiling* o perfilamiento busca definir grupos de clientes que poseen características similares. Un perfil o segmento de clientes puede basarse en atributos tales como datos demográficos, *hobbies*, canales de comunicación con la empresa preferidos, *likes* en redes sociales, patrones de compra o historial crediticio.

Asignar clientes a perfiles permite a la empresa hacer *targeting*, es decir, ofrecer productos o servicios específicos a los segmentos

deseados, además de «customizar»[2] las comunicaciones con sus clientes y tener una oferta de servicios consistente y focalizada.

## *Profiling: beneficios y consideraciones*

El perfilamiento de clientes es importante ya que «obliga» a la compañía a enfocarse en los diferentes segmentos de consumidores de interés, creando productos y servicios que sean atractivos y diseñando campañas de marketing focalizadas en estos grupos. Los beneficios para la empresa son los siguientes:

- Permite personalizar la oferta de servicios para determinados segmentos que sean de interés.
- Permite dimensionar los diferentes segmentos de clientes y lograr economías de escala, incluso teniendo un nivel importante de personalización.
- Permite mejorar la experiencia y satisfacción del consumidor al ofrecerles un trato personalizado.
- Permite dimensionar el o los segmentos objetivos en base a los registros de clientes, así como diseñar políticas de adquisición de clientes focalizadas y basadas en datos.
- Permite aumentar las ventas, maximizar utilidades e incrementar tasas de respuesta.
- Permite reducir la fuga de clientes en los segmentos de interés mediante campañas focalizadas de retención y programas de fidelización.

Existe una paradoja en los modelos de segmentación de clientes ya que estos permiten no solo incrementar la respuesta positiva de los clientes, sino también reducir el gasto innecesario en publicidad. Considerando, por ejemplo, el envío de cupones o catálogos, podemos tener una mayor respuesta de los clientes enfocándonos en quienes los datos nos dicen que podrían estar más interesados,

[2] Anglicismo. Adaptación al español de *customize* (personalizar).

a la vez que evitamos enviar «correos basura» *(spam)* a todos los clientes, con el costo de imagen que eso involucra.

Dentro de las acciones posibles asociadas a modelos de *profiling*, tenemos, por ejemplo, los programas de recompensa por fidelidad *(frequency reward programs)*. Estos programas buscan incrementar el valor del cliente al premiarlos de forma proporcional a sus compras acumuladas. Estos programas son comunes en sectores de servicios como hoteles, supermercados, restoranes, arriendo de automóviles o casinos. Las recompensas pueden ser productos gratuitos, devolución de dinero, descuentos o «puntos» en otro programa de fidelización que tenga la empresa.

La pregunta clave es si realmente este tipo de programas incrementa las utilidades de la empresa. La evidencia suele estar a favor de ellos. De todos modos, se deben tener en cuenta consideraciones importantes (Blattberg *et al.*, 2008):

- **Inmediatez**: se refiere al tiempo que pasa entre que se entrega la recompensa y el momento en que se puede utilizar. Algunos investigadores promueven el canje inmediato de los incentivos para fortalecer el efecto comportamiento-recompensa. Sin embargo, otros abogan por retrasar la entrega de la recompensa para que el cliente se enfoque más en la experiencia de marca por sobre el incentivo mismo.
- **Recompensa directa o indirecta**: una recompensa directa se refiere a un incentivo relacionado al producto mismo que lo gatilla o similar, mientras que uno indirecto puede ser un producto o servicio diferente, incluso dinero. Existe evidencia de que una recompensa directa genera un mayor compromiso e involucramiento por parte del cliente, mientras que otros investigadores afirman que no hay diferencias importantes entre ambos enfoques. Varios estudios se inclinan por recompensas en dinero; si bien suelen ser más costosas para la compañía, generan una mejor percepción del incentivo ya que el cliente puede utilizarlo de forma libre.
- **Valor monetario**: no es fácil definir el valor del incentivo, dado que existe una dispersión muy grande en oferta y descuentos. Obviamente, un valor alto del incentivo genera una mejor

impresión por parte de los consumidores y, por ende, un mayor efecto comportamiento-recompensa. Sin embargo, un incentivo de mucho valor puede reforzar la utilización de programas de recompensa más que la utilización del producto o servicio mismo. Lo mismo aplica a otros incentivos que no tienen un valor monetario claro.

- **Incentivo único o múltiple**: puede existir un incentivo único o una oferta amplia de recompensas que el cliente puede elegir, o bien la opción de recibir un incentivo varias veces. La desventaja de una solución de múltiples incentivos es el costo si mucha gente solicita el mismo incentivo, junto al efecto decreciente que puede tener el entregar el mismo incentivo varias veces al mismo cliente.
- **Incentivos utilitarios versus suntuosos**: existen estudios que sugieren que los consumidores prefieren incentivos suntuosos por sobre utilitarios cuando los requisitos del programa de recompensas son más restrictivos. Esto sugiere que la compañía puede ofrecer diferentes recompensas si cuenta con una estructura de incentivos compleja y basada en un sistema de umbrales, ofreciendo productos o servicios de lujo a clientes con un umbral alto y productos utilitarios para el resto.
- **Foco en posicionamiento de marca**: algunos estudios sugieren que las recompensas deben fortalecer la imagen de marca y que la intangibilidad de los incentivos es preferible para lograr esto. Por ejemplo, acceder a la página web por concursos y eventos puede generar una mayor lealtad que incentivos monetarios que puedan ser más tangibles.

Otra alternativa son los programas escalonados *(tier programs)*, que consisten en ofrecer diferentes productos o servicios dependiendo del segmento o eslabón en que se encuentren los clientes. Por ejemplo, una compañía puede definir cuatro eslabones —Bronce, Plata, Oro y Platino— y tratar a los clientes de cada grupo de manera diferente. Estos programas están ligados a los de recompensa ya mencionados, incluso el concepto de programas de lealtad se asocia a ambos. Sin embargo, existen algunas diferencias: los programas de recompensa suelen estar más enfocados a promociones específicas, como lo sería

una oferta de producto gratis. Por otro lado, un programa escalonado ofrece una diferenciación de largo plazo en base a la utilidad estimada que trae un cliente y genera un mayor compromiso con la empresa.

El proceso de generar perfiles de clientes considera la captura de datos sobre clientes, analizándolos para generar información sobre comportamientos, preferencias y otras características. Posteriormente es importante preparar la información de manera que se pueda presentar a los tomadores de decisiones para promover acciones focalizadas en mejorar la rentabilidad. Las diferentes características de los clientes permitirán combinar diferentes valores y establecer grupos que compartan comportamientos similares.

Dependiendo del objetivo de negocio que se quiera alcanzar, se deberá establecer el conjunto de variables que pueda representar de mejor manera el objetivo de negocio. Por ejemplo, si deseamos aumentar la lealtad y la retención de clientes, entonces variables relevantes a considerar pueden ser la edad del cliente y el tiempo asociado a la organización. Para aquellos clientes considerados como antiguos y cuyas ventas promedio se han reducido en un plazo de tiempo acotado, se podría ofrecer descuentos para recapturar.

De acuerdo con la literatura científica, los perfiles de clientes consideran tres fuentes de datos[3]:

- **Demografía**: datos asociados a edad, género, país de origen, estado civil, nivel de educación, nivel de ingresos o número de adultos en el hogar.
- **Psicología**: datos cualitativos que entregan un aspecto de la personalidad e identidad del cliente. Por ejemplo, si es feliz, sus creencias, intereses, estilos de vida.
- **Comportamiento**: generalmente asociado a datos históricos que reflejan patrones en común. Por ejemplo, el uso de algún producto, frecuencia de compra y volumen de compra, entre otros.

[3] https://www.commence.com/blog/2020/06/16/customer-profiling-methods/

## 2. Segmentación utilizando análisis RFM

En un contexto de recursos limitados, focalizar el esfuerzo de marketing hacia los clientes es crucial, identificar quiénes tendrán una mejor probabilidad de respuesta para materializar el esfuerzo y obtener resultados consistentes en el corto, mediano y largo plazo. Un análisis para mejorar la eficiencia de los esfuerzos de marketing es precisamente el análisis RFM, cuyas siglas en inglés significan *Recency* (cuándo fue la última compra de un cliente), *Frequency* (frecuencia de compras histórica) y *Monetary* (cuánto gasta un cliente, en promedio, en cada compra).

Este análisis se fundamenta en la historia de consumo de los clientes, considerando los datos de adquisición de productos o servicios en el pasado es posible predecir la propensión de reiterar esta transacción en un futuro. Una vez que se identifican quiénes son los clientes con más propensión a comprar, entonces se pueden efectuar esfuerzos de marketing focalizando tanto en el cliente, como en sus intereses de compra en particular (Blattberg *et al.*, 2008).

Generalmente el análisis establece una puntuación de 1 a 5, siendo 5 el puntaje más alto. En modelos más simples, se establece un promedio de los puntajes de los tres componentes RFM y se ordenan de mayor a menor, a modo de obtener aquellos clientes más valiosos. Otra forma es establecer ponderadores que permitan asignar más peso a alguno de estos tres componentes, teniendo en cuenta que la suma de los ponderadores debe ser igual a 1 (100%). A partir de estos puntajes también se construyen segmentos de clientes. El procedimiento va a depender de la combinación del puntaje de los tres componentes del análisis.

Por ejemplo, el segmento de clientes nuevos se puede caracterizar por una combinación de {5,1,X} dado que generaron una compra reciente, su *frequency* es bajo dado que es su primera compra y *monetary* depende de los datos (X representa cualquier valor entre 1 y 5). El segmento de clientes valiosos perdidos se puede caracterizar por una combinación de {1,X,5}, es decir, su última compra fue hace mucho tiempo, su *frequency* será un valor que depende de los datos y el *monetary* es alto. Los clientes más atractivos para la empresa y que conforman la masa crítica de ventas se caracterizan por una

combinación de {5,5,5}, es decir, compras recientes, alta frecuencia y altos montos. Claramente, estas combinaciones dependen del tipo de producto e industria. Por ejemplo, para la compra de automóviles tiene horizontes temporales mucho más extendidos que compras o consumo de servicios básicos, como agua embotellada o alimentos.

## *2.1 Definición y cálculo de los componentes del análisis*

Para la construcción del análisis, primero se debe comprender cada uno de los componentes y su forma de cálculo. El primer componente corresponde al *recency*, el cual considera cuándo fue la última compra de un cliente y quién ha comprado recientemente. Se fundamenta que un cliente con una compra reciente tendrá más fresca la idea de por qué adquirió el producto o servicio y su experiencia, en comparación con un cliente que realizó una compra hace mucho tiempo. Existe una posible relación inversa entre el *recency* y la probabilidad de compra: mientras más días pasen, es menor la probabilidad de respuesta a los esfuerzos de marketing. Para esto se puede establecer una fecha de análisis y una fecha límite. Generalmente se utiliza como fecha de análisis el cierre de mes más cercano, por otra parte, la fecha límite se establece un año atrás de la fecha de análisis. De esta forma tendremos el espectro completo de un año de compra. Para otras industrias puede variar dependiendo del tipo de producto, por ejemplo, para productos semidurables se encontró una relación en forma de U, esto debido a que en un plazo de un año o un poco más el cliente necesita reemplazar este producto. Por ejemplo, escobas, ítems de vestuario como poleras o calzado, cascos de motocicleta de resina termoplástica, entre otros. Posteriormente debemos obtener el número de días desde la última compra realizada hasta la fecha de análisis. En la tabla 1 podemos observar un ejemplo del enfoque RFM en donde el cliente N°1457 compró por última vez hace 307 días.

El segundo componente corresponde a *frequency*, que representa qué tan seguido el cliente ha adquirido el producto o servicio. Generalmente se contabiliza el número de transacciones, pero también se

puede presentar como el número de compras realizadas dividido en el tiempo que lleva un individuo siendo cliente. La relación entre lealtad y *frequency* es directa, ya que, si alguien adquiere el producto o servicio en reiteradas ocasiones, significa que tiene mayor preferencia por estos. En consecuencia, cualquier acción de marketing que se realice sobre este público tendrá una mayor probabilidad de respuesta de compra. Al igual que el caso de *recency*, su frecuencia y rango de tiempo para el análisis dependerá de la categoría del producto o servicio que se adquiera. En la tabla 1 nuevamente observamos al cliente N°1457 cuya frecuencia fue 297 veces, en tanto el cliente N°1460 lleva solo una compra en el horizonte de tiempo definido.

El tercer componente es *monetary*, que representa el valor monetario de las compras previas. Este componente se calcula directamente con la suma de las compras realizadas en la ventana temporal definida. De forma alternativa, se puede calcular como la suma de las compras dividida en el tiempo que lleva un individuo siendo cliente, o el promedio de los montos de compra. De igual forma que la frecuencia, es común encontrar una relación directa entre el valor monetario de las compras y la probabilidad de tener una respuesta positiva a un esfuerzo de marketing. En la tabla 1 observamos al cliente N°1457 cuyo monto total de compras asciende a $2.350.337, en donde se utiliza la primera forma de calcular *monetary*.

Tabla App1.1: Ejemplo de datos brutos para el enfoque RFM

| Cliente | Recency | Frequency | Monetary |
|---|---|---|---|
| 1457 | 307 | 297 | $2.350.337 |
| 1458 | 344 | 17 | $142.652 |
| 1459 | 339 | 20 | $301.488 |
| 1460 | 358 | 1 | $57.108 |
| 1461 | 153 | 3 | $287.418 |
| 1462 | 342 | 2 | $77.631 |
| 1463 | 338 | 19 | $184.273 |
| 1464 | 280 | 15 | $495.642 |
| 1465 | 96 | 108 | $1.031.970 |
| 1466 | 92 | 59 | $435.550 |

## *2.2 Confección del RFM Score*

Una vez que ya contamos con los datos brutos asociados al enfoque RFM, procedemos a tomar la distribución de cada uno de los componentes y dividirlo por quintiles. De esta forma, asignaremos al rango relevante el quintil correspondiente. Así, cada cliente tendrá un quintil asociado dependiendo de la magnitud de cada uno de los componentes calculados.

Tabla App1.2: Cálculo de quintiles RFM a partir de datos brutos

| Cliente | Recency | Frequency | Monetary | R | F | M |
|---|---|---|---|---|---|---|
| 1457 | 307 | 297 | $2.350.337 | 5 | 5 | 5 |
| 1458 | 344 | 17 | $142.652 | 5 | 3 | 3 |
| 1459 | 339 | 20 | $301.488 | 5 | 4 | 4 |
| 1460 | 358 | 1 | $57.108 | 5 | 1 | 1 |
| 1461 | 153 | 3 | $287.418 | 3 | 1 | 4 |
| 1462 | 342 | 2 | $77.631 | 5 | 1 | 2 |

Una vez que ya tenemos asignados los quintiles para cada componente, procedemos a construir el segmento RFM (de donde eventualmente se pueden obtener otro tipo de conjeturas) y el foco del análisis: el puntaje RFM. El segmento se confecciona concatenando cada uno de los quintiles de los componentes. El puntaje RFM se construye a partir de la suma de los quintiles de los componentes. Por ejemplo, el cliente N°1457 tendrá como puntaje la suma de R = 5, F = 5 y M = 5, en total 15 puntos.

Tabla App1.3: Cálculo del puntaje RFM a partir de quintiles

| Cliente | Recency | Frequency | Monetary | R | F | M | RFM Segmento | RFM Score |
|---|---|---|---|---|---|---|---|---|
| 1457 | 307 | 297 | $2.350.337 | 5 | 5 | 5 | 555 | 15 |
| 1458 | 344 | 17 | $142.652 | 5 | 3 | 3 | 533 | 11 |
| 1459 | 339 | 20 | $301.488 | 5 | 4 | 4 | 544 | 13 |
| 1460 | 358 | 1 | $57.108 | 5 | 1 | 1 | 511 | 7 |
| 1461 | 153 | 3 | $287.418 | 3 | 1 | 4 | 314 | 8 |
| 1462 | 342 | 2 | $77.631 | 5 | 1 | 2 | 512 | 8 |

Finalmente, podemos construir los segmentos de clientes a partir del puntaje RFM. Se recomienda observar la distribución del puntaje en términos de la cantidad de clientes y el promedio de los componentes para cada uno de los puntajes. A partir de este análisis se puede determinar, por ejemplo, que los clientes con puntaje mayor a 9 serán del segmento «Alto», aquellos entre 5 y 9 serán «Intermedio» y aquellos bajo 5 serán del segmento «Bajo». Es importante también conocer la conformación de estos segmentos, por ejemplo, tener en cuenta y presentar el número de observaciones y el promedio de los componentes RFM.

Tabla App1.4: Segmentos en base a RFM y valores promedio

| | | Recency | Frequency | Monetary |
|---|---|---|---|---|
| Segmento | N° Observaciones | Promedio | Promedio | Promedio |
| Alto | 13.674 | 201 | 42 | $460.357 |
| Bajo | 293 | 102 | 4 | $43.030 |
| Intermedio | 3.326 | 176 | 11 | $126.001 |

De forma alternativa, los datos brutos del enfoque RFM pueden servir como base para un modelo de *clustering*. Esta estrategia permite combinar la información RFM con variables sociodemográficas

(por ejemplo, edad, género, etcétera) u otras fuentes de información, con el fin de tener segmentos que representen una visión más completa del cliente. En este sentido, el enfoque RFM es una técnica de *transformación* (es decir, de preprocesamiento, ver capítulo anterior) que permite mejorar el poder predictivo de la información. Las técnicas de *clustering* se estudiarán en el capítulo siguiente.

## Referencias

Blattberg, R. C., Kim, B. -D. & Neslin, S. A. (2008). *Database Marketing: Analyzing and Managing Customers*. New York: Springer.

# 3. *Diagnostic Analytics*

En el capítulo anterior se describieron los conocimientos necesarios para realizar *descriptive analytics* a partir de data estructurada (tabular, es decir, almacenada en forma de tablas). Dicha área de la analítica tiene como objetivo obtener una imagen que represente, de forma sencilla y clara, la situación actual e histórica de una compañía.

Dentro del capítulo «*Descriptive Analytics*» se presentaron herramientas para estudiar y comprender las variables de interés de su compañía, de manera univariada (cada variable de manera individual); también se presentó una guía práctica para realizar análisis visuales de datos (a través de gráficos y tablas) y se explicaron metodologías utilizadas comúnmente para el preprocesamiento de su *data* (limpieza y transformación).

Un correcto análisis descriptivo de los datos permite obtener *insights* de manera sencilla, lo cual facilita a las juntas directivas, y altas gerencias de las empresas, la toma de decisiones. Adicionalmente, nos sirve para detectar errores en los datos. Sin embargo, su simpleza no permite identificar relaciones más complejas entre variables, por lo tanto, su aporte es limitado y está muy centrado en la capacidad del analista para tomar decisiones en base a los resultados del análisis.

Con *diagnostic analytics* damos un paso hacia la automatización de decisiones y la búsqueda de patrones complejos en los datos, comprendiendo las relaciones existentes entre las variables de interés y clasificando (segmentando) datos en grupos o perfiles de clientes. Esto se realiza con técnicas de *clustering* (Verbeke *et al.*, 2017).

*Diagnostic analytics* permite obtener *insights* a partir de sus datos, junto a información subyacente acerca de las relaciones entre sus variables de interés (por ejemplo, las ventas de un determinado producto en relación con la edad de sus compradores). Esta información

«escondida» dentro de sus datos se convierte en una nueva fuente de información para los departamentos de marketing, pensando en generar tanto nuevas propuestas de productos como promociones o estrategias de marketing focalizadas en un determinado grupo de interés (Sapp *et al.*, 2019).

Finalmente, *diagnostic analytics* y *predictive analytics* coinciden en el uso de los modelos supervisados transparentes, tales como regresiones y árboles de decisión. En economía, por ejemplo, es común utilizar modelos de regresión lineal o logística para testear hipótesis sobre los datos y, en particular, explicar un determinado fenómeno (variable dependiente) en base a otros factores (variables independientes). El modelo económico de Keynes es un caso muy conocido, que plantea que el gasto de consumo de los individuos depende (linealmente) de sus ingresos actuales. Un modelo como el de Keynes puede ser de *diagnostic analytics,* si ayuda a explicar un determinado fenómeno de forma multivariada (impacto de una o varias variables sobre otra), o de *predictive analytics,* si el objetivo es predecir el consumo de los individuos en base a su ingreso. En resumen, lo que define la taxonomía de las capacidades analíticas no es la técnica en sí, sino el objetivo del proceso de extracción de conocimiento de los datos. Estas técnicas transparentes de extracción de patrones serán estudiadas en el capítulo 6.

## 3.1 Inteligencia de negocios, visualización y *Dashboarding*

Es un largo camino el que deben recorrer las empresas para lograr una gestión eficiente de sus recursos disponibles. El uso de las herramientas de tecnologías de la información (TI) cobra aquí una importancia fundamental, ya que permiten realizar gestión, es decir, monitorear, gestionar y controlar los diferentes KPIs de interés.

Inteligencia de negocios (BI) corresponde a una aplicación práctica propia del área de análisis de datos a partir de la información empresarial, en la cual se «combina el análisis empresarial, minería de datos, visualización y herramientas para trabajar datos,

infraestructura y las mejores prácticas para ayudar a las organizaciones a tomar decisiones basadas en datos»[4].

Debido a la amplitud de técnicas que comprende el BI aplicado en las empresas, nos limitaremos a la visualización de datos y cómo esta es fundamental en la toma de decisiones.

La visualización avanzada basada en BI se ha convertido en un excelente punto de inicio en los procesos de transformación digital para las empresas. Esto se debe al valor agregado que genera y a que las tecnologías asociadas sean de fácil acceso y utilización.

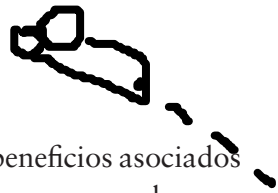
En términos prácticos, las visualizaciones poseen una serie de ventajas comparativas respecto a la construcción, validación e implementación de modelos predictivos. Algunas de estas ventajas se presentan a continuación:

- **Bajo costo de implementación**: un ejemplo claro de esto es el costo de la licencia del *software* de visualización *Power BI*, el cual es utilizado en gran parte del mundo empresarial. El costo es de tan solo US$ 10 por mes por usuario e incluye una amplia gama de funcionalidades puestas a disposición por Microsoft (incluye ciertos modelos predictivos básicos). Luego, si contrastamos el costo de *Power BI* contra un modelo predictivo que puede costar millones de pesos, se vuelve una opción muy atractiva.
- **Se requiere una baja preparación técnica**: otra gran ventaja de las herramientas de visualización de BI es respecto a los requisitos técnicos de capital humano necesarios para modelos de BI. En las empresas se requiere una baja preparación técnica, la cual, si no se posee, se puede adquirir con algunos cursos en línea de dificultad media-baja. Esto último se fundamenta en que no se requiere conocimientos de matemática ni de estadística compleja para obtener visualizaciones de datos de gran utilidad.
- **Cortos tiempos de desarrollo**: un tercer punto a favor que tienen las herramientas de visualización en contraste con la analítica predictiva es que los tiempos de desarrollo de los modelos de BI son realmente cortos. El producto final puede estar preparado

[4] *What is business intelligence? Your guide to BI and why it matters* (s. f.). Tableau https://www.tableau.com/learn/articles/business-intelligence

y finalizado en tan solo unas horas, siempre y cuando se tenga la información disponible y validada para corroborar los resultados indicados por el *dashboard* (herramienta de uso interactivo para los directorios de las empresas, gerencias o analistas involucrados en la toma de decisiones).

Después de entender las ventajas comparativas del uso de herramientas de visualización, conviene adentrarse en los beneficios concretos que generan para las empresas.



### *3.1.1 Beneficios para las empresas*

Una de las mejores maneras de explicar los beneficios asociados a la implementación de modelos de BI en las empresas puede ser a través de ejemplos concretos dentro de la industria; la implementación de *Dashboards*, es decir, interfaces gráficas de los modelos de BI a través de tableros interactivos, han demostrado que pueden generar aumentos de productividad, mayor eficiencia operacional, oportunidades de mejora, aumento en las ganancias y disminución en los costos para diversos tipos de empresas.

A continuación, tres ejemplos reales de empresas presentados por la empresa Tableau[5], otro grande de los *softwares* de visualización de datos a través de BI. Estas empresas integraron plataformas de inteligencia de negocios y han sido reconocidas a nivel internacional:

- ***Hellofresh***: corresponde a una empresa alemana de entrega de comida a domicilio que cotiza en la bolsa de Frankfurt, con sede en Berlín. Es el proveedor de *kits* de comida más grande de Estados Unidos, con operaciones en grandes capitales del mundo. Esta empresa puso en acción una solución de inteligencia de negocios centralizada que les permitió ahorrar entre 10 y 20 horas de trabajo por día, mediante la automatización de procesos de generación de informes. Además, el equipo de

[5] *What is business intelligence? Your guide to BI and why it matters* (s. f.). Tableau. https://www.tableau.com/learn/articles/business-intelligence

marketing principal pudo diseñar campañas de marketing digital regionales e individualizadas a través de modelos de BI.

- ***REI***: corresponde a una tienda de *retail* estadounidense minorista dedicada a la venta de artículos deportivos, equipo de *camping*, equipo de viaje y ropa. Posee más de 165 tiendas minoristas en 39 estados, también recibe pedidos por correo o Internet[6]. Sus ingresos anuales para el año 2019 fueron sobre los US$ 3.120 millones[7]. Esta empresa tenía el desafío de hacer un seguimiento de sus KPI para los clientes con datos de cercanos a los 90 terabytes de tamaño. Actualmente, la empresa utiliza una plataforma de BI para analizar sus clientes preferenciales (su membresía), la cual contribuye a más del 90% de las compras realizadas al minorista. Esencialmente, los KPI que siguen están relacionados a la adquisición, retención y reactivación de su membresía.
- ***Coca-Cola Bottling Company* (CCBC)**: esta empresa no necesita mucha presentación debido a su extenso tamaño y presencia mundial, como una de las empresas de expendio de bebidas gaseosas más grande del mundo. CCBC es la mayor embotelladora independiente de Coca Cola, la cual, gracias a la plataforma de BI, automatizó los procesos manuales de generación de informes, logrando «ahorrar más de 260 horas al año, es decir, más de seis semanas de 40 horas laborales»[8].

Con estos ejemplos nos podemos dar cuenta de la gran utilidad que tiene este tipo de herramientas a la hora de querer realizar seguimiento de la facturación de las ventas, la efectividad y rendimiento de sus campañas de marketing, la relación con los clientes, la planificación logística y el control de *stock*, entre otras múltiples tareas dentro del mundo empresarial.

---

6 «Informe de mayordomía de 2019». Equipo recreativo, Inc. 2020: https://www.rei.com/newsroom/article/rei-co-op-publishes-2019-full-year-financial-results

7 «REI Co-op publica los resultados financieros del año completo de 2019». rei.com. *Recreational Equipment*, Inc.27 de abril de 2020 https://www.rei.com/newsroom/article/rei-co-op-publishes-2019-full-year-financial-results

8 *What is business intelligence? Your guide to BI and why it matters* (s. f.). Tableau.

En el siguiente apartado, se presentará la metodología utilizada para el procesamiento de la información de los modelos de BI, llamada ETL por sus siglas en inglés *(Extract, Transform and Load)*, la cual entrega como producto final la información que alimenta los *dashboards* presentados a los tomadores de decisión.

### *3.1.2 Proceso de ETL: Extract, Transform and Load*

Como mencionamos anteriormente, el proceso de ETL consta de una serie de herramientas de extracción, transformación y carga utilizada, principalmente, para la extracción de datos desde distintas fuentes de datos, con el objetivo de limpiar, transformar e insertar data dentro de un almacén de datos (*datawarehouse* en inglés).

Veamos, a continuación, las tareas principales dentro de las tres etapas del proceso:

1. **Extracción de datos**: para esta etapa resulta fundamental familiarizarse con la información disponible para su modelo de BI. Para esto se realizan las conexiones correspondientes a planillas Excel, bases de datos relacionales (como PostgreSQL, SQL Server o MySQL), la nube (servicios como AWS, Google Cloud Platform o Microsoft Azure), sistemas de ERP (como SAP, Softland o Manager) o sencillamente mediante archivos planos. Parte de las tareas que debe realizar el analista de BI es establecer conexiones con las diferentes fuentes de datos y su consolidación.
2. **Transformación de datos**: esta etapa se puede subdividir en dos: limpieza de datos y transformación de datos propiamente tal. Esta diferenciación se suele realizar en el mundo del BI para enfatizar la importancia en la limpieza de los datos, ya que la calidad de los modelos de BI depende mayoritariamente de este proceso. Esto ocurre debido a la limpieza, que es fundamental para mostrar la información correcta y confiable, dado que permite encontrar una serie de posibles errores que podrían presentarse.

Errores comunes que se suelen encontrar en el proceso de limpieza de datos son errores de unicidad (es decir, considerar los valores únicos en caso de requerirse), errores de ortografía, redundancia (duplicidad), datos fuera de su rango *(outliers)*, errores de imputación y valores contradictorios. Luego se debe verificar la información obtenida del proceso de limpieza (se puede contrastar con la información mensual que se presentar en su *dashboard*, por ejemplo), se procede a la transformación de datos, la cual se dedica principalmente a estandarizar formatos, tratamiento de valores faltantes, eliminación duplicados e implementar ciertas restricciones en caso de errores dentro de la información futura. Por ejemplo, se pueden definir alertas para ser enviadas por correo electrónico que incluyan el detalle del error en la transformación correspondiente.

Además, se suelen hacer tareas más avanzadas como la fusión de datos a través de diferentes *joins* entre tablas, decodificar campos (mapear codificaciones de SKU por nombre de productos en *retails*, por ejemplo), construir *KPI*s a través de valores calculados y derivados. Ejemplos de *KPI*s, promedios mensuales acumulados de ventas o el cálculo del *stock* disponible para diferentes periodos.

Como ya es posible inferir, esta etapa está ligada con el preprocesamiento de los modelos KDD/CRISP-DM presentados en los capítulos pasados. La diferencia es el foco en *diagnostic analytics* del enfoque ETL, mientras que KDD/CRISP-DM se relacionan con la modelación predictiva.

3. **Carga de datos**: una vez que los datos fueron extraídos y transformados, estos se cargan en la fuente de datos final. En cuanto a la estructura de esta fuente de datos final, se explicará en la siguiente sección. En esta etapa se suelen utilizar tareas programadas tanto dentro de los servidores *on-premises* (locales) o dentro de las tecnologías ofrecidas por Cloud Computing («la nube»), para realizar el proceso de ETL con la frecuencia que solicite el analista o programador

encargado. Cabe mencionar que los procesos de ETL incluyen la comunicación del éxito o fracaso del proceso, ya que en múltiples ocasiones nos podemos encontrar con problemas en cambios de las fuentes de datos (por ejemplo, que el proveedor del sistema ERP realiza una actualización o algún departamento cambió alguna columna o tipo de dato de los *inputs* del ETL), por lo cual es necesario tener una automatización en los envíos de correo. Este envío automático de correos es ejecutado dentro del ETL.

### *3.1.3 Visualización de datos: ¿cómo obtener insights?*

Luego del proceso de preprocesamiento de datos a través del modelo ETL, nos encontramos en condiciones para abocarnos en la búsqueda de *insights* dentro de la información de su empresa. Dentro de los datos, están escondidos conocimientos importantes de utilidad para comprender y considerar antecedentes en la toma de decisiones. A estos conocimientos se les conoce, dentro del BI, como *insights*.

Para lograr obtener estos *insights,* resultan especialmente útiles las visualizaciones creadas a partir de la información procesada, es decir, cálculos de *KPI*s, métricas de interés, contraste de presupuestos contra gastos y toda aquella información relevante para la empresa. A partir de gráficos y tablas se pueden encontrar patrones subyacentes en los datos, conexiones dentro de la información, problemas de gestión de recursos en las empresas, pérdidas no identificadas anteriormente. Una buena definición de las ventajas de la visualización de datos nos la entrega Microsoft[9], a través de la siguiente explicación:

> La visualización de datos da vida a los datos y le permite descubrir y narrar los conocimientos ocultos en los números. A través de paneles en vivo, informes interactivos, diagramas, gráficos y otras representaciones visuales, la visualización de datos ayuda a los usuarios a desarrollar potentes conocimientos empresariales de forma rápida y eficaz.

[9] ¿Qué es la visualización de datos? | Microsoft Power BI (s. f.). *Ventajas de la visualización de datos*. https://powerbi.microsoft.com/es-es/data-visualization/.

Actualmente, en el mercado existen múltiples herramientas para visualización de datos, que pueden estar especialmente diseñadas para modelos comerciales, financieros, logísticos, de inversiones u otros. Los modelos más especializados pueden tener un costo monetario importante, por lo cual se vuelve atractiva la idea de utilizar herramientas como Power BI y Tableau. Alternativamente, existen otros proveedores, tales como QlikView, QlikSense, Amazon Web Services (AWS) y DataStudio de Google Cloud Platform (GCP). Estos modelos pertenecen a una gama de precios variable de acuerdo con cada plataforma, encontrándonos con precios de licencias por US$ 10 cada usuario por parte de Power BI, como también *softwares* con licencias de costos un poco más elevados como es el caso de Tableau o QlikSense, que cuestan US$ 30 o US$ 70 dependiendo de la versión del *software* que se compre.

El *software* Power BI es ampliamente utilizado por rubros muy variados debido a su bajo costo por usuario, como también por su comunidad activa y el rápido soporte proporcionado por Microsoft. Adicionalmente, es de fácil integración en las empresas con sistemas de esta marca, lo que reduce las barreras de entrada. Es posible también combinar con otros servicios ofrecidos por Microsoft 365 for Business, como lo es, por ejemplo, con Microsoft Azure *(Cloud Computing)* y con Microsoft SQL Server (*on-premises*).

Existen tutoriales «paso a paso» para cualquier usuario que quiera aprender conocimientos básicos-intermedios acerca de la creación de reportes, conexiones con fuentes de información, así como aprender a calcular métricas y columnas para utilizar en sus reportes. Plataformas como Datacamp, Coursera, Udemy o el mismo sitio de Power BI, entregan excelentes cursos (tanto gratuitos como pagados) para aprender todo lo necesario para construir modelos de BI que le permitan hacer gestión de su negocio y tomar decisiones en base a los datos que poseen.

Por último, hay una gran ventaja a la hora de aprender a crear reportes en Power BI en comparación con otros programas para BI: se puede utilizar el lenguaje de programación DAX (*Data Analysis Expressions*, por sus siglas en inglés) para realizar cálculos

de métricas más complejas. DAX tiene muchas similitudes en su sintaxis con las fórmulas utilizadas en Excel, por lo cual resulta sencillo familiarizarse con este lenguaje de programación.

## *3.2 Segmentación y Clustering*

Dentro de las tareas principales realizadas a través de *Diagnostic Analytics* se encuentra la segmentación o *clustering* de datos, la cual consiste en la división de un conjunto de observaciones (clientes, por ejemplo) en segmentos que posean la mayor homogeneidad posibles para cada categoría (que sea cohesiva), y, a la vez, posea la mayor heterogeneidad entre ellas (que logre distinguir características propias de cada conjunto y diferentes a las de los demás).

### *3.2.1 Motivación*

La segmentación posee una serie de aplicaciones relacionada al mundo empresarial, dentro de las cuales se encuentran las siguientes:

- Comprender una población de clientes, con la finalidad de caracterizarla y así poder realizar estrategias de marketing o publicidad dirigida.
- Asignar los recursos de marketing de manera eficiente, es decir, invertir en estrategias que tengan impacto en las ventas de una organización.
- Diferenciar las marcas pertenecientes a un portafolio.
- Identificar aquellos clientes más rentables.
- Identificar patrones de preferencia en las compras de los clientes.
- Identificar necesidades de los clientes, de manera de generar nuevos productos.

También existen algunas tareas interesantes que aún no han tenido gran desarrollo dentro del mundo de los negocios, tales como la segmentación de texto o de imágenes. Con respecto a lo primero, un ejemplo interesante es el enfoque utilizado por Jonathan Stray para minar Wikileaks, que nos sirve como motivación para esta sección.

Wikileaks, la organización sin fines de lucro fundada por Julian Assange, publica documentos filtrados y otras noticias de contenido sensible a través de su sitio web desde el año 2006. La mayor popularidad del sitio fue probablemente alcanzada con la filtración de los registros de la guerra de Irak *(Iraq War Logs)* el año 2010, la más grande filtración de documentos en la historia de los Estados Unidos.

El periodismo mundial está atento a cualquier filtración de datos. El objetivo es llegar lo antes posible con la noticia de mayor impacto, superando a la competencia. Si hay una filtración masiva de documentos en Estados Unidos, la prensa de un país como Chile puede simplemente buscar palabras clave en ellos, tales como «Chile», «Pinochet» u otros nombres relevantes. Sin embargo, ¿qué puede hacer un periodista estadounidense? Todas las comunicaciones hablan del país y, para el caso particular de los *Iraq War Logs*, «sumergirse» en 391.832 reportes militares parece una tarea titánica con el tiempo jugando en contra.

Afortunadamente, Jonathan Stray llegó con la respuesta a tiempo: *clustering* de documentos. Jonathan Stray podría ser considerado un «unicornio»: *Data Scientist* con vasta experiencia en aprendizaje de máquinas, pero a la vez periodista y editor en *Associated Press*. La metodología propuesta consiste en agrupar los documentos en base a su similitud, generando un mapa interactivo de la guerra de Irak en base a las palabras clave que se repiten en los diferentes documentos. Este trabajo se complementa con una herramienta de visualización de los grupos, sus etiquetas y la distancia entre los diferentes grupos. Esta herramienta le permite al periodista-investigador enfocarse en los eventos y aspectos de la guerra que desea estudiar, entregando, a la vez, una visión general del conflicto bélico[10].

### *3.2.2 Taxonomía y medidas de distancia*

Antes de formalizar las técnicas, nos gustaría hacer una distinción entre los conceptos «segmentación» y «*clustering*». El primero representa un enfoque general de agrupamiento, que puede ser basado en datos, o no. Por ejemplo, una segmentación en una empresa

[10] http://jonathanstray.com/a-full-text-visualization-of-the-iraq-war-logs

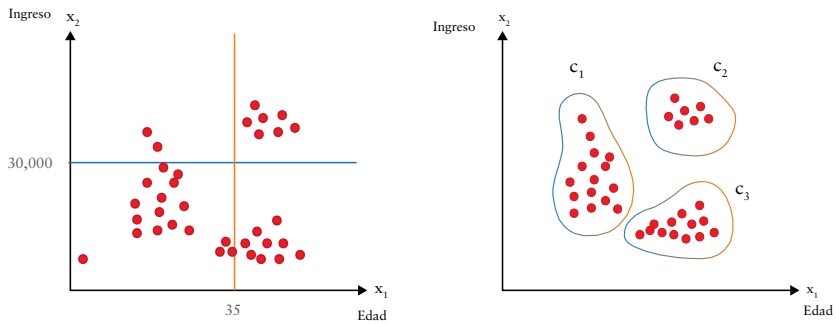
puede ser tan arbitraria como la siguiente: considerando las variables «edad» e «ingreso» y dos umbrales (35 años y US$ 30,000, ingreso medio en Estados Unidos), se construyen cuatro segmentos: «jóvenes de ingreso alto», «adultos de ingreso alto», «jóvenes de ingreso bajo» y «adultos de ingreso bajo».

Un primer problema con el enfoque anterior es que no considera la distribución de los datos, por lo que pueden quedar segmentos heterogéneos en tamaño que no representan la realidad de la empresa. Por ejemplo, un segmento como «jóvenes de ingreso alto» puede quedar subrepresentado frente a otros, perjudicando la accionabilidad de la estrategia. Un segundo problema es que esta estrategia no escala bien si queremos incluir nuevas variables. Si agregamos una variable como género, por ejemplo, ahora tenemos ocho grupos (hombres y mujeres de cada uno de los cuatro grupos). Esto termina agravando el primer problema.

El objetivo del *clustering* es encontrar similitudes y diferencias desde un punto de vista multivariado, es decir, calculando distancias entre datos considerando todas las variables a la vez. De esta manera podemos identificar la forma natural en que los datos se tienden a agrupar, haciéndonos cargo del primer problema, y seleccionando el número de segmentos adecuados con relación a la estructura de los datos, haciéndonos cargo del segundo problema.

Geométricamente hablando, la primera estrategia se puede representar en un plano cartesiano con cortes horizontales y verticales en cada variable, mientras que la segunda estrategia como una agrupación no-arbitraria basada en la estructura de los datos. Estas diferencias se ilustran en la figura 3.1.

Figura 3.1: Representación de *clusters* para un conjunto de clientes




Para realizar segmentación de datos, existen múltiples tipos de datos con los cuales se pueden crear categorías (*clusters*, de ahora en adelante) que definen aspectos comunes para cada una. Algunos ejemplos de tipos de datos para realización de *clusters* son: información demográfica, *lifestyle*, comportamiento histórico del cliente dentro de la empresa (patrones de compra, por ejemplo), investigación de mercado, redes sociales, etcétera (Baesens, 2014).

En términos matemáticos, las técnicas de segmentación de datos se basan en la idea de representar cada observación de la muestra de clientes como un punto dentro de un espacio multidimensional (espacio vectorial), donde cada una de sus dimensiones posee un valor numérico (binario, en el caso de las variables categóricas), lo que permite determinar distancia entre los puntos.

Las distancias entre observaciones se pueden cuantificar mediante el uso de normas vectoriales (o matriciales), las cuales permiten identificar aquellas observaciones más «similares» (que poseen menor distancia en espacio vectorial) e identificar sus características comunes.

En las ecuaciones 3.1 y 3.2 se pueden observar dos normas comúnmente utilizadas: la distancia Euclidiana y la distancia Manhattan (James *et al.*, 2017).

Ecuación 3.2

$$D_{Euclidiana}(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{n} (x_{i\,j} - x_{i',j})^2}$$

Ecuación 3.3

$$D_{Manhattan}(x_i, x_{i'}) = \sum_{j=1}^{n} |x_{i\,j} - x_{i',j}|$$

Con la finalidad de explicar el cálculo de las normas, se presenta el siguiente ejemplo: usted, como analista, desea identificar qué clientes poseen una mayor similitud en términos de las variables que lo definen (por ejemplo, edad, género, nivel de estudios y estado civil). Para ello requiere representarlos matemáticamente, con la finalidad de obtener una medida de «diferencia» entre ellos.

Siguiendo el razonamiento anterior, podemos calcular una norma para representar la diferencia entre estos dos clientes. Ahora bien, al cliente 1 lo llamaremos $x_i$ y al cliente 2 $x_{i'}$; ambos clientes tendrán n=3 variables que los representarán, las cuales serán la edad, el nivel de estudios y el estado civil. Estas características del cliente se identifican con el subíndice j en las ecuaciones de distancia.

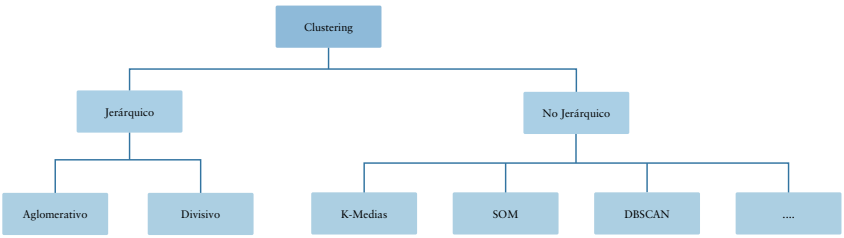
Finalmente, si tomamos una distancia a elección (Euclidiana, por ejemplo), luego solo restaría calcular las diferencias entre los valores de cada variable entre ambos clientes. El problema que encontramos aquí es que las variables cualitativas no se pueden restar directamente, a menos que las transformemos en variables numéricas: a las variables ordinales se les debe asignar una escala de valor incremental (por ejemplo, una persona con título universitario debe poseer un valor numérico mayor a una persona que haya terminado solamente estudios de educación media), y las variables nominales deben convertirse en variables *dummies*, es decir, variables binarias donde el valor «1» significa que posee cierta característica y «0» que no la posee).

El concepto de distancias entre datos es la base para entender por qué las técnicas de *clustering* permiten segmentar los datos en

grupos de clientes con determinadas características comunes. Cabe mencionar que el cálculo de distancias entre observaciones son una parte común para todas las técnicas de *clustering*, por lo cual se realiza en todos esos algoritmos.

En la figura 3.2 se indica una clasificación para técnicas de *clustering*, la cual comprende dos grandes grupos: modelos jerárquicos y no jerárquicos. Los modelos jerárquicos se pueden subdividir a su vez en estrategias aglomerativas y divisivas. Por su parte, existe una amplia variedad de estrategias no jerárquicas, en donde destacamos K-Medias *(K-means)* como una de las más utilizadas en negocios. Alternativamente, *self-organizing map* (SOM) es un modelo que corresponde a la versión no supervisada de las redes neuronales, mientras que DBSCAN es una estrategia popular para buscar valores extremos *(outliers)* identificando zonas densas (muchos ejemplos muy cercanos entre sí) y zonas menos densas en los datos (Baesens *et al.*, 2015).

Figura 3.2: Tipos de *Clustering*




### *3.2.3 Clustering jerárquico*

Dentro de las técnicas de *clustering* jerárquico, existen dos opciones principales a partir de las cuales se desprenden técnicas más sofisticadas, estas son el *clustering* jerárquico aglomerativo y *clustering* jerárquico divisivo.

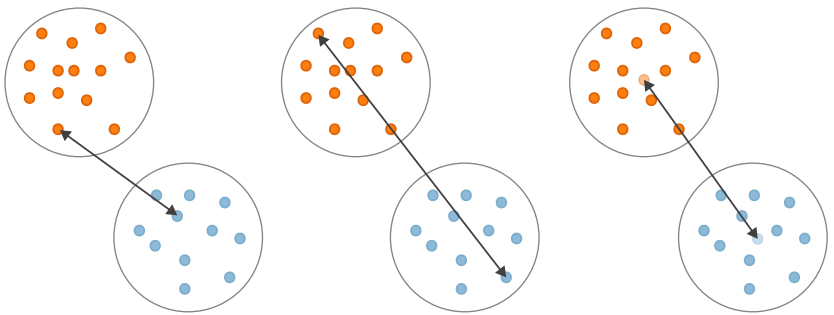
Las técnicas de *clustering* jerárquico divisivo comienzan a partir del conjunto de datos completo como un solo grupo, que luego se

divide en grupos más pequeños hasta llegar a formar tantos grupos como individuos tiene la muestra.

En cambio, las técnicas de *clustering* jerárquico aglomerativo funcionan de manera inversa, es decir, comienzan con $n$ grupos (las $n$ observaciones de manera individual), para ir fusionando los grupos con características más similares hasta formar un grupo grande que termine incluyendo todos los datos.

Un aspecto importante del método es que no es trivial representar la distancia que existe entre un grupo y otro. Si consideramos que un *cluster* es solo un ejemplo, la distancia entre grupos es directa ya que utilizamos este punto como representante del *cluster*. Sin embargo, cuando un grupo está formado por varias observaciones, debemos elegir un representante para el cálculo de la distancia. Lo más común es utilizar alguna de estas tres alternativas: enlace simple, enlace completo y enlace promedio. Estos enfoques se pueden visualizar en la figura 3.3 (Han *et al.*, 2011):

Figura 3.3: Cálculo de distancias entre *Clusters*



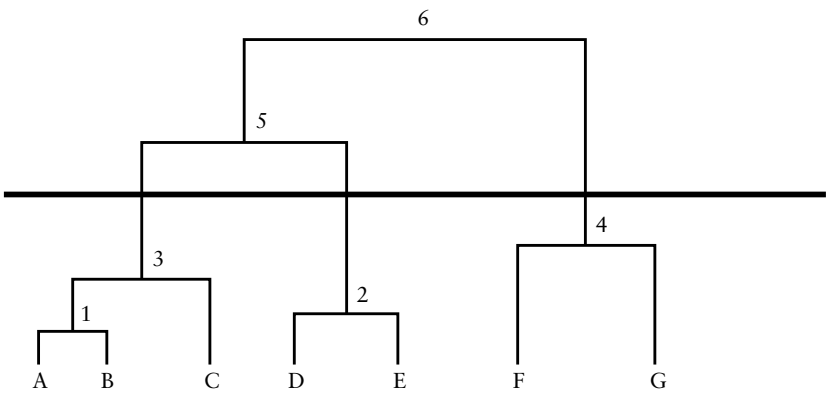

- **Enlace simple** *(Single Linkage)*: define la distancia entre dos *clusters* como la distancia más corta posible entre dos ejemplos, uno de cada *cluster*.

- **Enlace completo** *(Complete Linkage)*: define la distancia entre dos *clusters* como la mayor de las posibles distancias entre los elementos de ambos *clusters*.
- **Enlace promedio** *(Average Linkage)*: calcula el promedio de todas las posibles distancias entre todos los ejemplos de un *cluster* hacia los ejemplos del otro. Este tipo de enlace suele ser preferido por sobre los anteriores, ya que logra un mejor balance entre la distancia más corta y la más larga.

Una de las principales ventajas de los modelos jerárquicos por sobre la mayoría de los modelos no jerárquicos es que el número de *clusters* puede ser definido en base al mismo algoritmo y no de forma arbitraria. Para determinar la cantidad de grupos existen varios criterios basados en el «costo» en términos de distancia, que implica fusionar o separar dos grupos.

Una opción gráfica para decidir el número óptimo de *clusters* es la elaboración de un dendrograma (ver figura 3.4). Un dendrograma corresponde a un diagrama con forma de árbol que registra una secuencia de fusiones. Esta representación puede ser horizontal o vertical (como en la figura 3.4). Cada número representa las fusiones de grupos, interpretando el dendrograma de abajo hacia arriba.

Figura 3.4: Dendrograma para análisis de *Clustering*



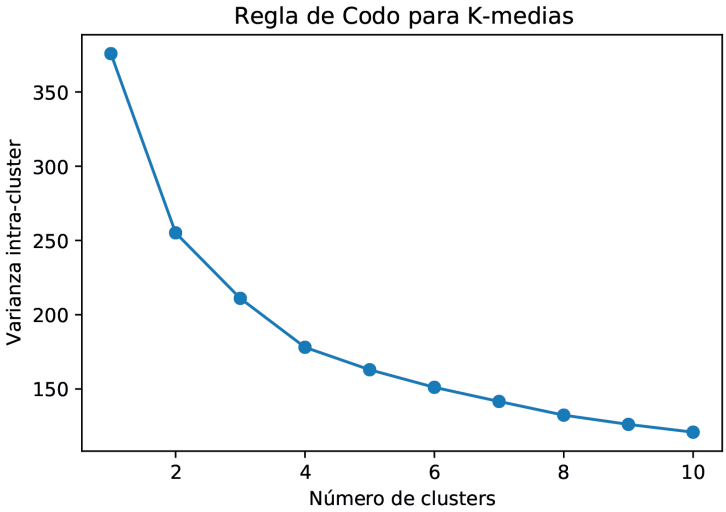
Para determinar el número de *clusters* óptimo en el ejemplo, se debe trazar una línea horizontal, como se muestra en la figura 3.4. Comenzando desde abajo (todos los elementos separados), mientras más subimos la línea, menos grupos tendremos. Sin embargo, llegará un punto en que el costo de pasar de *k* a *k*-1 *clusters* sea demasiado alto. Aquí es donde decidimos quedarnos con *k clusters*.

Volviendo a nuestro dendrograma, la línea horizontal se encuentra ubicada justo entre la cuarta y la quinta fusión. Esta solución sugiere tres *clusters* (A,B,C - D,E - F,G). La distancia (vertical) entre la tercera y la cuarta fusión es relativamente pequeña, al igual que con las primeras fusiones. Sin embargo, la distancia entre la cuarta y la quinta fusión es mucho mayor en términos relativos. De acuerdo con esto, el número óptimo de *clusters* es tres.

Una desventaja del dendrograma es que se vuelve bastante difícil de interpretar en la práctica, con centenas o millares de ejemplos. Una segunda alternativa para determinar el número de *clusters* óptimo es la utilización de un *scree plot* o gráfico de sedimentación, el cual permite representar la relación entre número de *clusters* con la distancia entre estos.

Por medio de un *scree plot,* se puede realizar la misma metodología utilizada con el dendrograma para determinar la cantidad de *clusters* óptimo, siendo de más fácil interpretación cuando nos enfocamos solo en las fusiones que nos llevan a menos de 10 *clusters*. En la figura 3.5 se puede observar un ejemplo de un *scree plot,* donde el eje X representa el número de *clusters* mientras que el eje Y la distancia que «cuesta» pasar de un número de grupos a otro. La medida más común es la varianza intra-*cluster,* es decir, la dispersión total que existe entre todos los elementos que pertenecen a un *cluster.*

Figura 3.5: Scree plot




### *3.2.4 Clustering no jerárquico*

Como se mencionó anteriormente, las técnicas de *clustering* no jerárquico permiten realizar una segmentación de las observaciones con estrategias de diversa índole.

Dentro de las técnicas más utilizadas en el *clustering* no jerárquico se encuentra *K-Means* (K-Medias), que corresponde a una estrategia de particionamiento que establece *a priori* la cantidad de grupos en que se dividirá la *data*. La principal razón de su popularidad en aplicaciones de analítica de negocios radica en su simpleza y sus excelentes resultados. En particular, es menos sensible a los *outliers* que los métodos jerárquicos, entregando grupos que son representativos en términos de cantidad de objetos. Esto es muy importante en el perfilamiento de clientes, donde buscamos grupos *accionables*, es decir, ser capaces de aplicar una política de negocios a cada uno de los diferentes segmentos encontrados. Los métodos jerárquicos suelen identificar grupos muy pequeños en cantidad de datos que contienen principalmente valores extremos, lo que afecta negativamente la validez de estos *clusters* al momento de decidir una acción comercial.

El algoritmo de *K-Means* es un procedimiento iterativo que comprende los siguientes pasos:

1. (Inicialización) Seleccionar al azar las $K$ observaciones que servirán como centroides (centros de *clusters*) a partir de las cuales se formarán las $K$ agrupaciones.
2. Asignar cada observación al *cluster* que tenga el centroide más cercano a la observación. Para esto se calcula la distancia de cada ejemplo a cada uno de los $K$ centroides, y se identifica la menor distancia.
3. Recalcular la posición de los $K$ centroides como el promedio de las ubicaciones de cada ejemplo en el *cluster* respectivo.
4. Repetir los tres pasos anteriores hasta que los centroides dejen de cambiar (o su cambio se vuelva marginal).

El elemento clave dentro de este algoritmo es el número de *clusters* $K$, el cual debe ser especificado antes de comenzar el análisis. Para determinar el número de *clusters* es posible construir un *scree plot* como el de la figura 3.5. En este caso, eso sí, se requiere la construcción de varios modelos de $K$ medias con un número creciente de $K$. Como medida de desempeño del modelo (representada en el eje Y del gráfico) se suele considerar la «suma de cuadrados dentro del *cluster*» (*within-cluster sum of squares* o WCSS). Para obtener esta medida, primero se calcula la distancia Euclidiana al cuadrado (sin calcular la raíz cuadrada en ecuación 3.1) entre cada punto y el centroide al cual este objeto fue asignado. Posteriormente se suman todas estas distancias dentro de un mismo *cluster* y se divide por la cantidad de observaciones en el *cluster*. Este proceso se repite para cada *cluster*. El valor de WCSS es el promedio del cálculo anterior para todos los *clusters*.

La medida WCSS representa la dispersión promedio que existe dentro de cada *cluster*. Una configuración de *clustering* con mayor dispersión promedio que otra implica que la solución es peor. Sin embargo, mientras más grande sea el $K$, menor será la dispersión necesariamente. En el caso extremo, el óptimo es siempre que cada

ejemplo sea un *cluster* único, llegando a WCSS=0. Entonces, ¿cómo podemos elegir la mejor solución?

La respuesta se encuentra en la *regla de codo* descrita en el dendrograma de la figura 3.4. No buscamos el WCSS mínimo, sino la **variación marginal** mayor. Siguiendo el ejemplo del dendrograma, si el «costo» de pasar de tres a dos *clusters* es significativamente mayor que los costos anteriores (de cuatro a tres, de cinco a cuatro, etcétera), deberíamos quedarnos con K=3 como valor óptimo.

Finalmente, es importante reconocer que el paso 1 del algoritmo K-medias consiste en una inicialización al azar de los centroides. Este paso puede causar cierta inestabilidad ya que cada vez que se ejecuta el algoritmo, este puede llegar a soluciones diferentes. Se recomienda entonces ejecutar K-medias con diferentes semillas aleatorias (números utilizados para inicializar la generación de valores al azar), de manera de verificar la estabilidad de la solución una vez que el valor de $\overline{K}$ óptimo haya sido seleccionado.

## *3.3 Reglas de asociación*

Actualmente, la oferta de productos y servicios ha aumentado considerablemente, lo cual ha generado un nivel de competencia muy alto dentro de rubros como el *retail,* las instituciones bancarias, telecomunicaciones y la tecnología. Dentro de este contexto, las empresas requieren enfocar sus esfuerzos, en términos de los recursos utilizados en marketing (como se mencionó en la sección 3.2), los cuales permiten llegar con las ofertas adecuadas tanto a los clientes antiguos, como a los clientes nuevos.

Debido a esta problemática, surge la necesidad de identificar patrones comerciales en las preferencias de los clientes por determinados productos que suelen adquirir en conjunto, lo cual puede ser abordado mediante reglas de asociación (Baesens, 2014; Verbeke *et al.*, 2017).

Las reglas de asociación corresponden a recomendaciones de compras para clientes en base a sus compras anteriores (o las de otros clientes parecidos a ellos) y suelen construirse a partir de las bases

de datos transaccionales, donde cada transacción se compone de un identificador de la compra (ID) y un conjunto de ítems $\{i_1, i_2, i_3, \ldots, i_n\}$ (por ejemplo, productos comprados en un supermercado) perteneciente al conjunto total de ítems $I$.

En la tabla 3.1 se puede observar un ejemplo de un conjunto de datos de transacciones históricas que pueden ser asociada a uno o múltiples clientes:

Tabla 3.1: Ejemplo de Dataset de transacciones

| ID | Ítems |
|---|---|
| 1 | Queso, jamón, tomate, leche |
| 2 | Bebida, queso, pan, tomate, jamón |
| 3 | Jamón, tomate, lechuga |
| 4 | Pan, queso, jamón, tomate |
| 5 | Bebida, leche, pan, tomate |
| 6 | Carne, bebida |
| 7 | Pan, yogurt, leche, queso, jamón |
| 8 | Leche, helado, pan |
| 9 | Tomate, lechuga, manzanas |
| 10 | Queso, pan, leche, jamón, tomate |

Formalmente, una regla de asociación se puede ver como una regla de lógica de la forma $X \rightarrow Y$, donde $X \subset I$, $Y \subset I$, y $X \cap Y = \emptyset$. $X$ identifica al antecedente de la regla, mientras que $Y$ es la consecuencia de la regla. Algunos ejemplos de reglas de asociación son:

- Si un cliente compró torta y globos, entonces con un 80% de probabilidad querrá comprar serpentinas.
- Si un cliente compró leche, queso y jamón, entonces con un 90% de probabilidad querrá comprar mantequilla.
- Si un cliente compró televisor, pilas y computador, entonces con un 70% querrá comprar un equipo de sonido.

Cabe mencionar que, como se pudo observar en los ejemplos anteriores, las reglas de asociación son probabilísticas por naturaleza, es decir, son medidas estadísticas que permiten afirmar, con un cierto nivel de confianza o probabilidad de ocurrencia, que dicha asociación, finalmente, se cumplirá.

Para medir el grado de certeza que tienen las reglas de asociaciones se utilizan las medidas *support* (o soporte) y *confidence* (o confianza), las cuales son dos medidas claves para cuantificar la fuerza de una regla de asociación. También existe la medida de *Lift,* la cual busca el mismo objetivo de *support* y *confidence,* pero considerando también la probabilidad de que la regla se de en la base de datos solo por el azar.

### *3.3.1 Support*

En un conjunto de transacciones puede haber reglas que suenen muy interesantes, pero pueden ser tan infrecuentes que no tengan validez estadística. Por ejemplo, pensemos que en un supermercado el 95% de los clientes que llevan caviar ruso llevan whisky de 18 años también. Un 90% de confianza. Suena extremadamente alto, ¿no? Pero si nos damos cuenta de que solo diez personas han comprado caviar ruso en el lapso de todo un mes (y, por ende, nueve de ellas llevaron whisky de 18 años), terminaremos por concluir que la regla es demasiado «exótica» como para ser considerada accionable. No tiene sentido diseñar promociones o ajustar las góndolas para una regla donde solo 10 personas han participado.

¿Cómo podemos filtrar este tipo de reglas? El *support* de un determinado conjunto de ítems es definido como el porcentaje total de transacciones dentro de una base de datos que contiene ese conjunto de ítems. De esta manera, la regla $X \rightarrow Y$ posee un *support* del 100% si el 100% de sus transacciones en el conjunto de datos contiene $X \cup Y$.

El *support* de un determinado conjunto de ítems puede ser definido mediante la siguiente fórmula:

Ecuación 3.4

$$support(X \cup Y) = \frac{número\ de\ transacciones\ que\ contiene\ (X \cup Y)}{número\ total\ de\ transacciones}$$

La asociación $\{queso, jamón\} \rightarrow \{tomate\}$ tiene un soporte de $4/10$ o 40% ya que cuatro de las diez transacciones contienen queso, jamón y tomate. Esta regla puede considerarse como bastante frecuente en base a los productos que la conforman, siendo potencialmente atractiva para el diseño de acciones comerciales.

Es importante notar que el soporte es simétrico, es decir, las reglas $\{queso, jamón\} \rightarrow \{tomate\}$ y $\{tomate\} \rightarrow \{queso, jamón\}$ tienen el mismo soporte.

### *3.3.2 Confidence*

Una regla con productos que se llevan de forma frecuente no la hace necesariamente interesante. El valor de una regla radica en la relación entre el antecedente y la consecuencia. Esto se mide con el concepto de confianza.

La *confidence* de un determinado conjunto de ítems es definido como la razón entre el *support* de la regla de asociación en cuestión $(X \cup Y)$ sobre el *support* del antecedente de la regla $(X)$.

Ecuación 3.5

$$Confidence(X \rightarrow Y) = P(X) = \frac{Support(X \cup Y)}{Support(X)}$$

La *confidence* mide la fuerza de la asociación y se define como la probabilidad condicional de la consecuencia de la regla $(Y)$ dado el antecedente de la regla $(X)$.

Volviendo al ejemplo de la tabla 3.1, la regla de asociación $\{queso, jamón\} \rightarrow \{tomate\}$ tiene un soporte de $4/10$ y el antecedente de la regla posee un soporte de $5/10$, por lo tanto, la confianza sería $8/10$ u $80\%$. Otra forma de verlo es que cuatro de las cinco transacciones que llevaron queso y jamón también llevaron tomate ($4/5$=80%). Las transacciones 1, 2, 4 y 10 tienen tanto queso y jamón como tomate, mientras que la transacción 7 no incluye este último ítem.

A diferencia del soporte, la confianza no es simétrica, es decir, las reglas $\{queso, jamón\} \rightarrow \{tomate\}$ y $\{tomate\} \rightarrow \{queso, jamón\}$

no tienen necesariamente la misma confianza. Queda como ejercicio para el lector realizar el cálculo de la confianza para esta regla.

### *3.3.3 Lift*

Es bastante común que se generen reglas relevantes (con soporte y confianza altos) en base a productos muy populares. El pan, por ejemplo, es un producto que casi no puede faltar en las canastas chilenas. Con un producto así es común que se formen muchas reglas del tipo «producto X -> pan», que pueden ocurrir simplemente por azar.

El *lift,* también conocido como medida del interés, toma en consideración el aspecto de la probabilidad previa de la consecuencia de la regla ($Y$):

*Ecuación 3.6*

$$Lift(X \rightarrow Y) = \frac{support(X \cup Y)}{support(X)\ support(Y)}$$

Un valor de *lift* cercano o menor que 1 indica que una regla es «peligrosa», en el sentido de que puede ocurrir al azar porque el soporte de la consecuencia es muy alto. Volviendo al ejemplo de la tabla 1, la regla $\{queso, jamón\} \rightarrow \{tomate\}$ tiene un *lift* de 2: 0,4 / (0,4*0,5). Este es un *lift* bastante alto y lejano de 1, por lo que otorga la tranquilidad que la regla no se debe al azar.

### *3.3.4 Algoritmo a priori*

La construcción de reglas de asociación se hace en base al algoritmo *a priori* (o variantes más modernas), lo que le otorga el carácter de estrategia de minería de datos (no supervisada). El algoritmo consiste en dos pasos que se aplican de forma iterativa: (1) identificar todos los grupos de ítems que son frecuentes (tienen un soporte mayor a un umbral dado) y (2) descubrir todas aquellas reglas de asociación que tienen confianza alta (alternativamente se puede utilizar *lift*).

La creación de reglas es un problema combinatorial de alta complejidad, ya que existen muchas formas diferentes de combinar los productos en una base (incluso si filtramos por los frecuentes) para llegar a reglas nuevas. Sin embargo, el problema se facilita enormemente al considerar un principio bastante simple: cada subconjunto de un conjunto de ítems infrecuentes es infrecuente por definición.

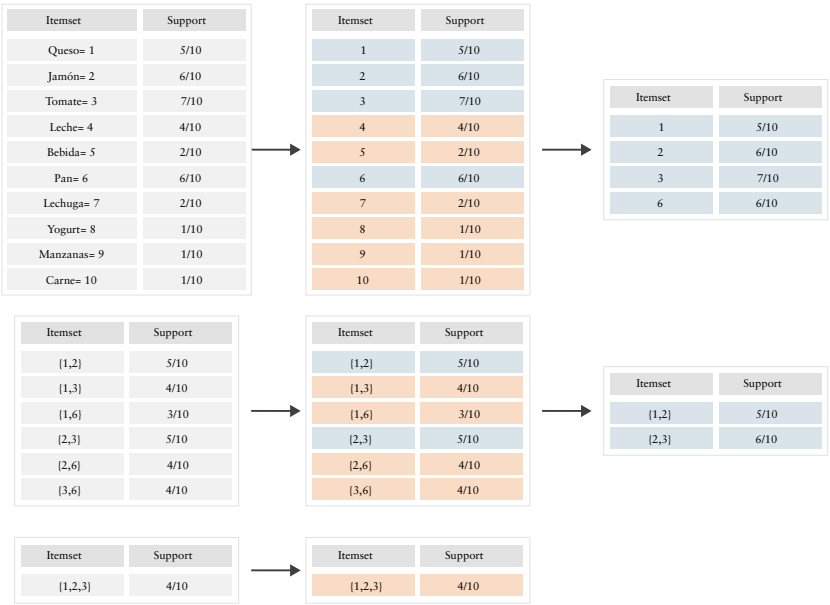
El algoritmo sigue la siguiente lógica. Se comienza con k=1 (productos únicos) y se filtran aquellos no frecuentes. Luego se define k=k+1 y se construyen grupos de productos de tamaño *k* uniendo pares de conjuntos de elementos frecuentes con *k-1* ítems y eliminando aquellos conjuntos que tiene subconjuntos poco frecuentes. Gracias a esta propiedad, se puede disminuir el número de subconjuntos candidatos a evaluar, lo cual mejora sustancialmente el rendimiento del algoritmo, debido a que requerirá realizar menos operaciones en la base de datos.

A modo de ejemplo, consideremos la tabla 3.1 con $minsup = 0.5$ como umbral de soporte mínimo. El algoritmo *a priori* se presenta en la figura 3.6.

El algoritmo *a priori* genera como salida una lista de ítems frecuentes, que pueden ser de uno o varios productos. De acuerdo con la figura 3.6, el algoritmo encontró siete ítems: cuatro de un producto, tres de dos productos y uno de un producto. En el siguiente paso se construyen reglas de la forma $X \rightarrow Y$ en base a combinaciones de estos ítems.

Para la construcción de reglas, es posible generar la combinatoria de ítems y luego ordenar las reglas generadas de mayor a menor de acuerdo con *confidence* o *lift*. De esta forma nos quedarán las reglas más atractivas primero, las cuales podemos utilizar para el diseño de promociones, generar recomendaciones personalizadas o incluso proponer combinaciones diferentes de productos en las góndolas.

*Figura 3.6: Ejemplo algoritmo* a priori




### *3.3.5 Collaborative filtering*

El filtrado colaborativo (CF) corresponde a una técnica utilizada por los sistemas de recomendación, los cuales tienen dos objetivos principales: uno predictivo y otro de filtrado de información.

En primer lugar, CF busca realizar predicciones automáticas sobre los intereses de un usuario basado en las preferencias o gustos de muchos usuarios, que se consideran de manera colaborativa. El supuesto fundamental de este enfoque es la siguiente consideración: «si una persona A tiene la misma opinión que una persona B sobre un tema, entonces es más probable que A tenga la misma opinión de B sobre un tema diferente que la de otra persona escogida al azar».

En este sentido, los gustos de una persona podrían servir para predecir qué películas, por ejemplo, le gustarían a otra, dada una lista de los gustos de esa segunda persona. No se debe olvidar que las predicciones son propias para cada usuario, pero que sí utilizan información obtenida de muchos usuarios.

En segundo lugar, tenemos el filtrado colaborativo como un proceso de identificación de patrones, o filtrado de información, que utiliza la colaboración de múltiples agentes, fuentes de datos y opiniones. De esta manera, las aplicaciones de filtrado colaborativo requieren grandes cantidades de datos *(*Big Data*)* y se basan principalmente en técnicas de minería de datos mediante modelos de *deep learning* (ver capítulo 8).

## Metodología

Los sistemas de CF tienen variadas formas, pero muchos de los sistemas comunes se pueden sintetizar en dos pasos, que se basan en la idea «los usuarios que compraron X también comprarán Y», considerando el ítem que se desea recomendar como elemento central del análisis:

1. Construir una matriz ítem-ítem que determine las relaciones entre cada par ordenado de ítems, de modo de identificar con un valor «1» aquel par de películas X e Y que vio.
   Inferir los gustos del usuario al que deseo recomendar, de acuerdo con la matriz del paso 1.
2. En otras palabras, la idea es buscar coincidencias de las preferencias del usuario activo (aquel a quien quiero recomendar) con aquellas preferencias de otros usuarios que anteriormente hayan seleccionado las mismas preferencias que el usuario activo.

## Referencias

Baesens, B. (2014). *Analytics in a Big Data World*. New Jersey: John Wiley and Sons.

Baesens, B., Van Vlasselaer, V. & Verbeke, W. (2015). *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*. New Jersey: John Wiley and Sons.

Gülagiz, F. K. & Sahin, S. (2017). «Comparison of hierarchical and non-hierarchical clustering algorithms». *International Journal of Computer Engineering and Information Technology*, 9(1), 6.

Han, J., Kamber, M. & Pei, J. (2011). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Sapp, C., Brabham, D., Antelmi, J., Cook, H., Craig, T., Barot, S., Galli, D., Pal, S., Mohan, S. & Gilbert, G. (2019). *2019 Planning Guide for Data and Analytics*. https://www.gartner.com/en/doc/361501-2019-planning-guide-for-data-and-analytics

Vassiliadis, P., Simitsis, A. & Skiadopoulos, S. (2002). «On the logical modeling of ETL processes». In *International Conference on Advanced Information Systems Engineering* (pp. 782-786). Berlin: Springer.

Verbeke, W., Baesens, B. & Bravo, C. (2017). *Profit Driven Business Analytics*. New Jersey: Wiley.

# Laboratorio 2: Segmentación de clientes y *clustering*

En este laboratorio segmentaremos la base de crédito que preprocesamos anteriormente. La idea es aplicar K-medias a la base para encontrar perfiles de clientes que nos permitan tomar decisiones dentro del banco.

Primero importamos las librerías a utilizar (aquí se muestran solo algunas librerías; el detalle completo está en el sitio web del libro):

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
import plotly.express as px
```

El siguiente paso es montar Google Drive para poder enlazar nuestra base de datos. De esta forma podemos cargar después nuestra base de datos creditSIM.xlsx (que debemos tener en una carpeta «Colab Notebooks» en nuestro Drive claramente identificada).

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Cargamos la base preprocesada del Laboratorio 1:

```
ruta_data_set = 'gdrive/My Drive/creditSIM_preprocess.xlsx'
data_set = pd.read_excel(ruta_data_set)
```

Como preprocesamiento, solo escalaremos los datos entre 0 y 1. Esto es requisito para los modelos de *clustering*.

```
del data_set[«VarObj_S»] scaler = MinMaxScaler()
scaler_data = scaler.fit(data_set)
data_set_scaled = pd.DataFrame(scaler_data.transform(data_set), index=data_
set.index, columns=data_set.columns)
```

## 1. K-medias 1: regla de codo

La regla de codo permite identificar un rango de valores para k, el número de *clusters* que debemos fijar *a priori* antes de ejecutar K-medias.

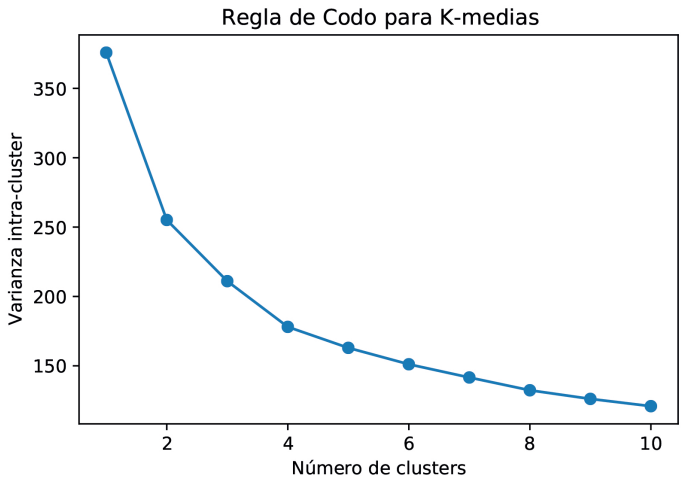
Primero hacemos un ciclo *for* de 1 a 11 *clusters* para encontrar el número de grupos adecuado para nuestros datos, luego calculamos la varianza intra-*cluster* (wcss) para cada configuración:

```
wcss = []
for i in range(1, 11):
kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
kmeans.fit(data_set_scaled)
wcss.append(kmeans.inertia_)
```

Entonces graficamos esta varianza intra-*cluster* obtenida (eje Y) para cada número de *clusters* (eje X):

```
plt.plot(range(1, 11), wcss, 'o-')
plt.title('Regla de Codo para K-medias')
 plt.xlabel('Numero de clusters')
plt.ylabel('Varianza intra-cluster')
plt.show()
```

Figura 1: Gráfico regla de codos para K-medias



Podemos ver en el gráfico que de 1 a 4 *clusters* disminuye significativamente la varianza, pero luego logra estabilizarse. El número adecuado de grupos parece estar entre 4 y 6 *clusters,* que es donde se forma el «codo» del gráfico.

## 2. K-medias 2: solución con cuatro *clusters*

Creamos la segmentación con K medias fijando el número de *clusters* a considerar (4) y definimos una semilla (825) para asegurar que cada vez que corramos el modelo, tengamos los mismos resultados. El comando *kmeans.fit* realiza la construcción del modelo sobre la base preprocesada.

```
kmeans = KMeans(n_clusters=4,random_state=825)
kmeans.fit(data_set_scaled)
```

Antes de interpretar cada grupo para decidir qué acciones tomar (y, en base a esto, elegir el número de *clusters* adecuado), debemos asegurarnos de que los *clusters* tengan un tamaño representativo. Grupos muy pequeños no son accionables.

A continuación, se muestra el tamaño de cada *cluster,* en donde el comando *bincount* realiza un conteo o tabla de frecuencias para cada etiqueta en el objeto *labels*.

```
labels = kmeans.labels_
counts = np.bincount(labels[labels>=0])
size_clusters = pd.DataFrame(counts, columns=['size'])
print(size_clusters)
```

```
size
0       796
1       936
2       546
3       722
```

Vemos que el *cluster* más pequeño tiene 546 clientes (18.2%), lo que es bastante representativo. La interpretación de cada *cluster* viene dada por los centroides, que pueden considerarse como un elemento representativo de cada grupo.

A continuación, cada fila corresponde a un centroide y se entrega el valor en cada atributo imprimiendo en pantalla los *kmeans. cluster_centers_* como un *dataframe*.

```
centers = kmeans.cluster_centers_
centers = pd.DataFrame(centers, columns=data_set.columns)
print(centers)
```

```
 AgnosDirec AgnosEmpleo Edad .. Ln_Ingreso2 Ln_DeudaExt Ln_DeudaInt
0   0.046956   0.046990 0.212975 ...   0.393653   0.419021   0.359559
1   0.111882   0.138798 0.257620 ...   0.570857   0.603484   0.534474
2   0.255299   0.366690 0.513754 ...   0.661396   0.682124   0.602695
3   0.092866   0.122709 0.533398 ...   0.472854   0.484062   0.401893
[4 rows x 7 columns]
```

Para poder analizar los centros de los *cluster* tenemos que desnormalizarlos, ya que al ser obtenidos con datos normalizados no son fáciles de interpretar. Utilizaremos el comando *scaler.inverse_transform* que está diseñado precisamente para retornar los valores originales de un conjunto de datos normalizado.

Además de los centros desnormalizados, agregaremos una nueva columna ID al final para identificar cada centroide en un gráfico. El comando de Numpy *arange* genera valores equidistantes entre valores de un determinado rango y nos proporciona un ID con valores 0-1-2-3 para cada uno de los cuatro grupos.

```
centersde = scaler.inverse_transform(centers)
data_centers = pd.DataFrame(centersde, columns=data_set.columns)
data_centers['ID'] = np.arange(len(data_centers))
print(data_centers)
```

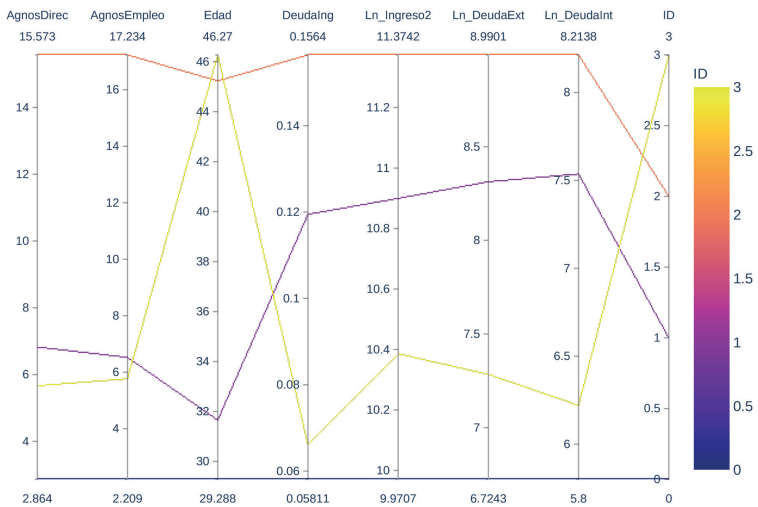
```
AgnosDirec AgnosEmpleo   Edad ... Ln_DeudaExt Ln_DeudaInt ID
0    2.864322    2.208543  29.287688 ...    6.724304    5.800025  0
1    6.824786    6.523504  31.653846 ...    8.312849    7.536546  1
2   15.573260   17.234432  45.228938 ...    8.990083    8.213835  2
3    5.664820    5.767313  46.270083 ...    7.284422    6.220313  3
[4 rows x 8 columns]
```

Una vez obtenidos los centros desnormalizados, realizamos un gráfico de coordenadas paralelas para interpretar los centroides. En este gráfico se ilustra cada variable dentro del modelo de *clustering*

como líneas verticales. Las líneas horizontales representan cada centroide, con un valor en cada atributo, e ID entrega el color de la línea. De esta forma, podemos visualizar las diferencias entre los centroides y transformar un problema de visualización de múltiples dimensiones, en uno de solo dos.

```
fig = px.parallel_coordinates(data_centers, labels=data_centers.columns,
color='ID')
fig.show()
```

Figura 2: Gráfico de coordenadas paralelas




Del gráfico anterior tenemos que:

- El *cluster* 0 (azul) tiene valores pequeños en todas las variables, representando a clientes jóvenes con bajo ingreso y deuda. Podemos ofrecer a estos clientes un plan especial de «cuenta joven» ya que, si bien no son clientes valiosos ahora, tienen el potencial de crecer a futuro.

- El *cluster* 1 (violeta) también agrupa a jóvenes (en promedio, 32 años), pero estos tienen un nivel de endeudamiento mucho más alto. Este segmento es claramente más riesgoso y, por ende, podemos tomar acciones de «*damage control*» en el sentido de evitar pérdidas futuras con ellos.
- El *cluster* 2 (bermellón) reúne a los «*heavy users*», es decir, los que tienen valores altos en todas las variables. Son clientes maduros y valiosos, por lo que podemos tomar acciones para mantenerlos contentos y tratarlos como clientes *premium*.
- El *cluster* 3 (amarillo) es bastante interesante ya que también corresponde a clientes de edad más madura, pero con un muy bajo endeudamiento. Se ven clientes más bien seguros, por lo que podemos ofrecerles ventas cruzadas y *upselling*, con el objetivo de que crezcan con la empresa.

## 3. K-medias 3: solución con cinco clusters

La solución anterior es muy razonable en términos de cantidad de clientes por *cluster* y acciones que podemos realizar en base a cada grupo. Veamos qué pasa si agregamos un nuevo *cluster* (n_*clusters*=5):

```
kmeans = KMeans(n_clusters=5,random_state=825)
kmeans.fit(data_set_scaled)
```

A continuación, se muestra el tamaño de cada *cluster:*

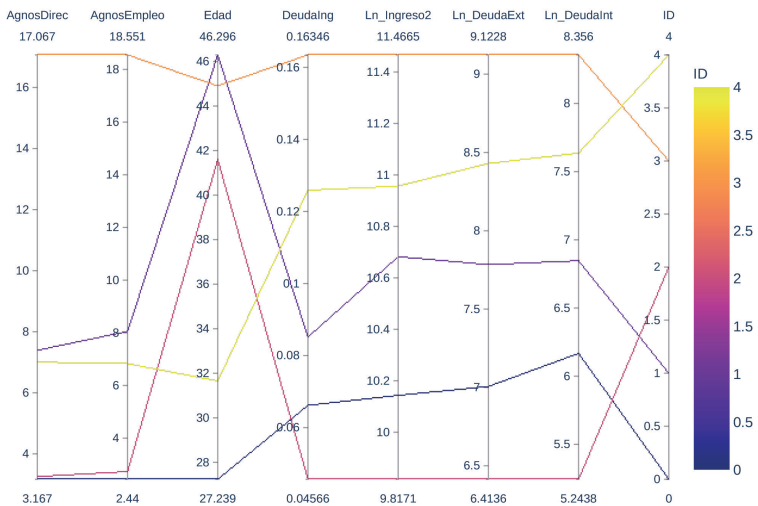
```
labels = kmeans.labels_
counts = np.bincount(labels[labels>=0])
size_clusters = pd.DataFrame(counts, columns=['size'])
print(size_clusters)
```

```
   size
0   712
1   633
2   428
3   432
4   795
```

Vemos que el *cluster* más pequeño tiene 428 clientes (14%), lo que sigue siendo representativo. Vamos ahora al gráfico de coordenadas paralelas:

```
centers = kmeans.cluster_centers_
centers = pd.DataFrame(centers, columns=data_set.columns)
 centersde = scaler.inverse_transform(centers)
data_centers = pd.DataFrame(centersde, columns=data_set.columns) data_
centers['ID'] = np.arange(len(data_centers))
fig = px.parallel_coordinates(data_centers, labels=data_centers.columns,
color='ID')
fig.show()
```

Figura 3: Gráfico de coordenadas paralelas




Del gráfico anterior tenemos que:

- El *cluster* 0 (azul) corresponde a clientes jóvenes, pero ahora tienen un mayor endeudamiento que en el ejemplo pasado. El patrón es más claro con cuatro *clusters*, pero sigue siendo válido el enfoque «cuenta joven».

- El *cluster* 1 (morado) agrupa a clientes de edad más avanzada y endeudamiento relativamente bajo. El patrón es más claro con cuatro *clusters,* pero se sugiere el enfoque de ventas cruzadas (*cluster* amarillo del gráfico anterior).
- El *cluster* 2 (rojo) incluye a clientes de edad avanzada y endeudamiento bajo. Este grupo nace de una separación de los *cluster* azul y amarillo de la solución con cuatro grupos. La acción adecuada es las ventas cruzadas.
- El *cluster* 3 (naranjo) tiene a los «*heavy users/premium*» de la solución anterior.
- El *cluster* 4 (amarillo) junta a los «*damage control*» de la solución anterior.

Da la impresión de que las cuatro acciones quedaban definidas de mejor forma con la configuración anterior de cuatro *clusters* y, por ende, recomendamos esta solución con cuatro acciones claramente definidas. De todos modos, es posible seguir y diseñar acciones para cada grupo, teniendo siempre cuidado con que el grupo más pequeño no tenga un número muy bajo de clientes.

# 4. Fundamentos de las bases de datos

En este capítulo hablaremos de los aspectos fundamentales de las bases de datos, pasando por su diseño, necesidad, gestión y utilidad. Veremos las técnicas más clásicas del modelado de bases de datos para luego revisar estrategias acordes a la magnitud de datos que se manejan hoy.

## 4.1 Bases de datos y su diseño

La información está en todas partes y viene en distintos volúmenes y formatos. La forma que más nos sirve a quienes la estudiamos son los datos: una representación concreta de la información que se puede almacenar y procesar.

Los datos son esenciales en la sociedad desde que la tecnología comenzó a ganar terreno en el mundo y, con ellos, las bases de datos han cumplido un rol fundamental a la hora de manejar estos grandes volúmenes: desde su almacenamiento hasta su procesamiento. Para las empresas, el manejo de los datos es una parte fundamental de su operación, por lo que, en este apartado veremos la estructura y el diseño de una base de datos para hacernos una idea sobre las bases de datos y su funcionamiento.

## *4.1.1 Fundamentos y arquitectura de una base de datos tradicional*

### *4.1.1.1 Fundamentos*

Anteriormente introdujimos las bases de datos hablando de su importancia, pero ahora debemos formalizar este y otros conceptos.

Una base de datos puede ser definida como una colección de datos pertenecientes a un negocio específico o a un problema (Lemahieu, vanden Broucke & Baesens, 2018). Ejemplos de esto puede ser el sistema de datos de una tienda que contiene productos, vendedores y sucursales, también un colegio, donde existen alumnos que asisten a clases y cada clase tiene un profesor y una sala asignados.

Una base de datos no se gobierna sola; es necesario tener un sistema que se encargue de gestionarla, así como esta se encarga de gestionar los datos. Por ello, también definimos el sistema de manejo de base de datos (*DBMS* por sus siglas en inglés). Es un *software* usado para definir, crear, utilizar y mantener una base de datos. La combinación de una base de datos y un DBMS hacen un sistema de base de datos (Lemahieu, vanden Broucke & Baesens, 2018). Existen diversas DBMS, tales como Oracle o MS-SQL server. MySQL es un DBMS muy conocido y de código abierto.

Adicionalmente, los sistemas de base de datos son manejados por los usuarios mediante un lenguaje de manipulación de datos (DML por sus siglas en inglés). Más adelante hablaremos en detalle de este concepto, pero por ahora es bueno quedarse con la definición.

#### *4.1.1.1.1 Modelo de datos*

Si bien los sistemas de bases de datos sirven para almacenar estos últimos, necesitamos poner orden en este «almacén». Para esto debemos contar con un modelo de datos, que nos ayude a entender de mejor manera el sistema al cual se adecuan los datos guardados.

Reconoceremos tres tipos de modelado de datos:

1. El **modelo conceptual**, que provee una descripción a alto nivel de los datos, con sus características y relaciones (Lemahieu, vanden Broucke & Baesens, 2018). De esta forma se facilita la comunicación entre el arquitecto de información y el usuario del negocio.
2. El **modelo de datos lógico**, que es la traducción del modelo conceptual a un ambiente en el cual se tengan que implementar los datos de alguna forma. Este modelo aún puede ser entendido por el usuario del negocio, ya que no es lejano a él (Lemahieu, vanden Broucke & Baesens, 2018). Más adelante estudiaremos los tipos de modelos que existen y cómo podemos hacer una buena elección.
3. El **modelo de datos externo**, que contiene varios subconjuntos de datos en el modelo lógico, también llamadas vistas, ajustadas a las necesidades de aplicaciones o usuarios en específico (Lemahieu, vanden Broucke & Baesens, 2018).

#### *4.1.1.1.2 Datos estructurados y no estructurados*

Es importante mencionar que los modelos previamente descritos solo se pueden aplicar a datos estructurados. Este tipo de datos era el único que existía en los tiempos de los primeros sistemas de bases de datos. En estos se pueden expresar y formalizar los atributos de los datos, así como el nombre o la dirección de una persona. La principal ventaja de este tipo de datos es la integridad: nos aseguramos de que siempre existan datos y no tengamos vacíos en las entradas.

Lo anterior no es posible para una base de datos no estructurada, en donde los datos no tienen una forma estándar y, por ende, no se pueden definir atributos o condiciones que deban cumplir todos los datos. Generalmente, los datos no estructurados tienen un volumen mayor que el de los estructurados. Debido a esto, tienen también un mayor potencial para extraer información valiosa. Sin embargo, hay que saber almacenarlos y procesarlos adecuadamente. Más adelante aprenderemos los modelos existentes para estos tipos de datos y cómo podemos almacenarlos.

*4.1.1.1.3 Ventajas de los DBMS*

Los DBMS tienen muchas ventajas como la anteriormente descrita de poder manejar datos estructurados. Entre ellas se encuentran (Lemahieu, vanden Broucke & Baesens, 2018):

1. **La independencia de datos**: los cambios en las definiciones de los datos tienen muy poco impacto en las aplicaciones que usan los datos.
2. **El modelado de la base de datos**: como anteriormente mencionamos, los datos pueden ser modelados y definidos, en especial sus características y relaciones.
3. **El manejo de redundancia de datos**: poder manejar los casos en los cuales se repitan datos y poder establecer reglas para esto; poder establecer reglas de integridad, enfocadas en mejorar la corrección de los datos, cómo deben ser representados y almacenados.
4. **El control de concurrencia**: los DBMS pueden manejar la ejecución paralela de distintos programas que usan la base de datos, dando foco en el buen rendimiento.

Por último, tenemos la seguridad de los datos y algunas utilidades de rendimiento para las operaciones de la base de datos.

*4.1.1.2 Arquitectura*

Manejar un sistema de base de datos no es sencillo, estos necesitan tener una arquitectura establecida para gestionar todas las operaciones, tales como almacenamiento, gestión y consulta de los datos. En este apartado definimos las principales partes de un DBMS.

*4.1.1.2.1 Gestor de conexión y seguridad*

El gestor de conexión se encarga de facilitar la conexión con la base de datos, verificando las credenciales de ingreso tales como usuario y contraseña. Luego de esto, retorna una conexión con la cual podemos conectarnos a la base de datos y utilizar todas sus

funcionalidades. Por otro lado, el gestor de seguridad se encarga de revisar los permisos de escritura o lectura que tenga un usuario y asegurarse de que se cumplan al momento de revisar una solicitud de acción de un usuario. Por ejemplo, algunos usuarios solo deberían ser capaces de leer datos, mientras que otros pueden leer y escribir.

*4.1.1.2.2 Compilador DLL*

El compilador de biblioteca de enlace dinámico (*DLL* por sus siglas en inglés), está encargado de compilar las definiciones dadas a la base de datos. Traduce estas definiciones de manera que puedan ser gestionadas por la base de datos. Una vez que son compiladas correctamente, se genera un catálogo para el DBMS que contiene estas definiciones de datos para poder ser usadas por el resto de los componentes.

*4.1.1.2.3 Procesador de consultas*

Es de las partes más importantes de un DBMS, pues se encarga de gestionar las consultas a la base de datos tales como lectura, inserción, actualización y eliminación de datos. Está conformado por el compilador DML, Analizador de Consultas, Reescritor de Consultas, Optimizador de Consultas y el Ejecutor.

*4.1.1.2.4 Gestor de almacenamiento*

Maneja el acceso a los datos y supervisa el uso eficiente del almacenamiento en la base de datos. Consiste en el Administrador de Transacciones, Administrador de Búfer, Administrador de Bloqueo y el Administrador de Recuperación. Sus componentes se encargan de manejar desde la ejecución de las consultas hasta la revisión de la correcta ejecución de las transacciones.

*4.1.1.2.5 Utilidades del DBMS*

Esto es externo a la arquitectura fundamental de un DBMS, consiste en varias utilidades que entregan funcionalidades a veces

esenciales para los usuarios de una base de datos. Algunas utilidades comunes son de carga, que sirve para importar datos desde otras bases de datos o archivos de texto; utilidad de reorganización, que reorganiza los datos para mejorar la eficiencia; utilidades de rendimiento, que nos entregan indicadores del rendimiento de nuestra base de datos; utilidad de gestión de usuarios, para gestionar usuarios, grupos de usuarios y sus privilegios y, finalmente, una utilidad de copia de seguridad y recuperación.

#### *4.1.1.2.6 Interfaces del DBMS*

Una DBMS necesita tener herramientas que le permitan a los usuarios poder interactuar y utilizar todas las funcionalidades disponibles. Por esto existen interfaces, hechas por los proveedores de estos sistemas. MySQL es uno de los proveedores más conocidos de base de datos y provee una herramienta que podemos ver en la figura 4.1.

### *4.1.2 Tipos de DBMS*

A continuación, presentamos algunos de los tipos de bases de datos más utilizados por los DBMS. Estos difieren en elementos como su arquitectura, capacidad de acceso en simultáneo y utilización.

#### *4.1.2.1 Jerárquicas*

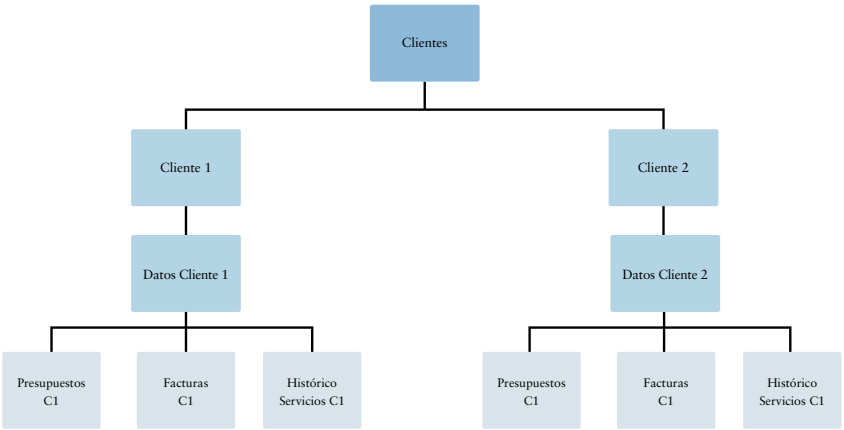
Utiliza jerarquías mediante la representación de una figura de un árbol invertido. Es uno de los primeros modelos creados. Las jerarquías se implementan siguiendo el concepto de padre e hijo, en el cual, los padres contienen y conocen a los hijos. Es importante considerar que los padres tienen muchos hijos, pero cada hijo solamente tiene un padre, como se muestra en la figura 4.1. Algunos ejemplos de DBMS que ocupan este tipo son IMS de IBM y Registry de Microsoft Windows.

Sus principales componentes son los tipos de registro y los tipos de relaciones. Los tipos de registro son un conjunto de registros que

describen entidades similares (Lemahieu, vanden Broucke & Baesens, 2018). Puede tener cero o muchos registros y cada uno posee atributos específicos, definidos por el tipo de registro.

El tipo de relación se encarga de unir dos tipos de registro, donde solo se permiten relaciones jerárquicas en las que el padre tenga muchos hijos, es decir, una relación 1 a N. En el caso de la figura 4.1, la entidad clientes tiene clientes los cuales tienen datos y estos datos tienen detalles como los presupuestos, las facturas y el histórico de los servicios.

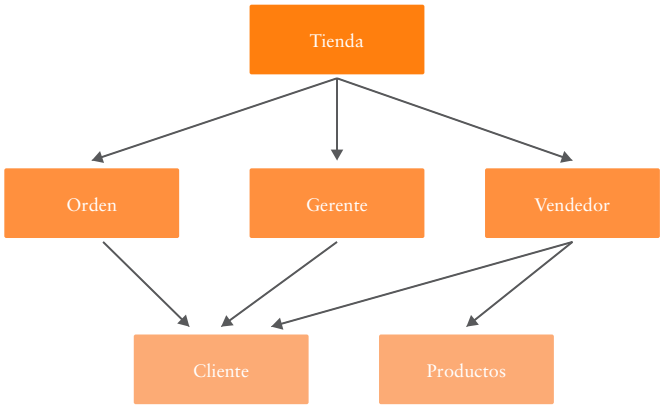
Figura 4.1: Esquema de un DBMS jerárquico




### *4.1.2.2 Redes*

Es más flexible que las bases de datos jerárquicas, permitiendo representar los objetos y sus relaciones como un grafo, tal como se muestra en la figura 4.2, en donde se modela las relaciones entre las entidades de una tienda. El ejemplo más conocido es el CODASYL DBMS, que implementa el modelo CODASYL.

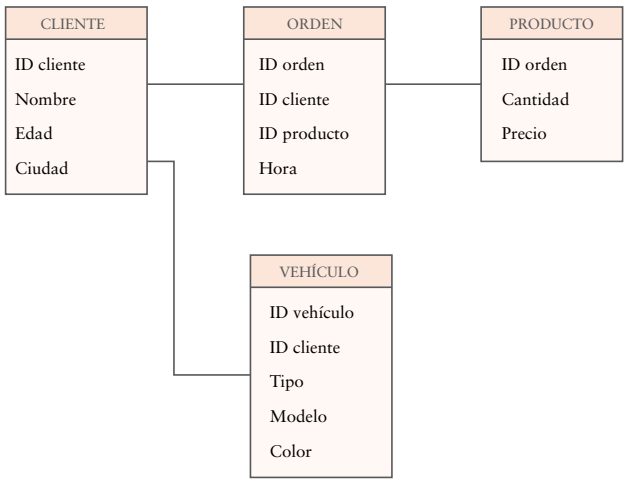
Figura 4.2: Esquema de un DBMS de redes




### *4.1.2.3 Relacionales*

Es el modelo más utilizado y reconocido por la estructura de relaciones que forman sus componentes. Típicamente utiliza SQL como DML. Es usado principalmente por la independencia de datos que provee gracias a la separación entre los modelos lógicos e internos (Lemahieu, vanden Broucke & Baesens, 2018). Los ejemplos más conocidos son MySQL, Oracle, Microsoft SQL Server y DB2 de IBM. En la figura 4.3 podemos ver cómo se modelan estas relaciones, dictadas principalmente por atributos en común entre dos entidades. Por ejemplo, una orden tiene el ID de un cliente.

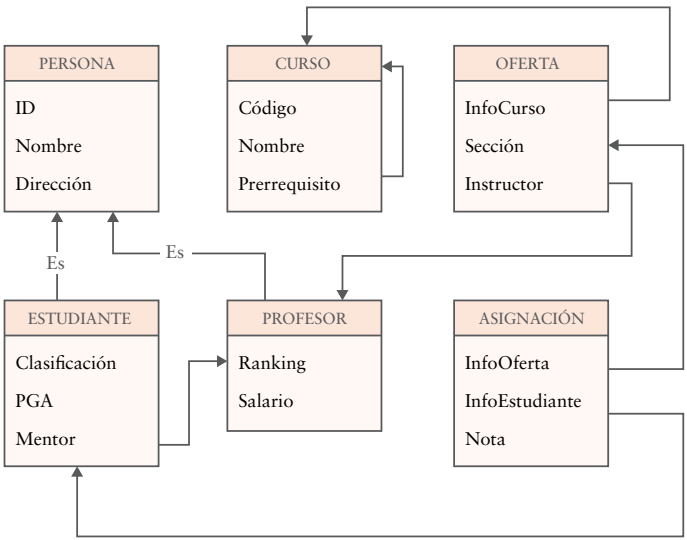
Figura 4.3: Esquema de un DBMS relacional



### *4.1.2.4 Orientadas a objetos*

Este tipo de DBMS utiliza conceptos de orientación a objetos como tipos definidos, funciones definidas, colecciones y herencia. Es muy parecido al modelo relacional, por lo que algunos DBMS como Oracle, DB2 y Microsoft SQL Server lo utilizan. En la figura 4.4 se puede ver su similitud con el modelo relacional. La principal diferencia es que las relaciones del DBMS orientado a objetos no consideran un atributo en común que dicte una relación.

Figura 4.4: Esquema de un DBMS orientado a objetos




### *4.1.2.5 XML*

Utilizan el modelo *XML* para guardar los datos, el cual es un estándar. Su representación es parecida a una base de datos jerárquica. Es un buen modelo para intercambiar datos entre aplicaciones debido a su naturaleza. En el siguiente ejemplo, en las figuras 4.5 y 4.6, podemos ver su diagrama de árbol y una instancia de este.

Figura 4.5: Esquema DBMS XML

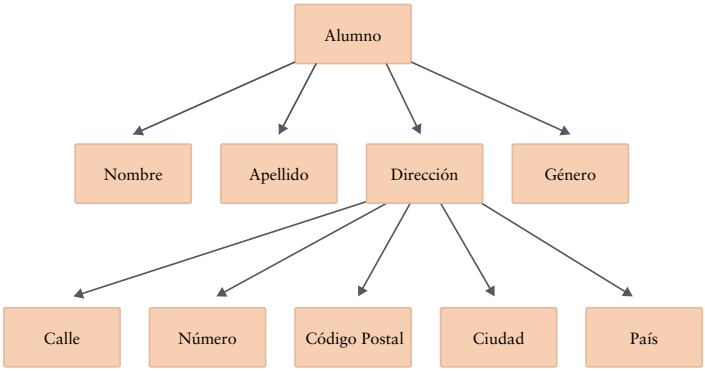



Figura 4.6: Instancia de un elemento XML

```
<alumno>
      <nombre>Carla</nombre>
      <apellido>Vairetti</apellido>
      <dirección>
            <calle>calleNombre</calle>
            <número>87</número>
            <zipcode>7500000</zipcode>
            <ciudad>7500000</ciudad>
            <país>Chile</país>
      </dirección>
            <género>mujer</género>
</alumno>
```

### *4.1.2.6 NoSQL*

Es el modelo más importante hoy en día, por lo que lo veremos en profundidad más adelante. Su nombre viene del término en inglés *not-only* SQL. Algunos ejemplos más típicos son MongoDB o Neo4j. Generalmente se dividen en cuatro categorías: orientada a documentos, llave-valor, tuplas o columnas. Su principal ventaja es que son muy escalables, a cambio de su falta de estructuración y alta volatilidad. Más adelante veremos en detalle los distintos tipos de modelos SQL que existen.

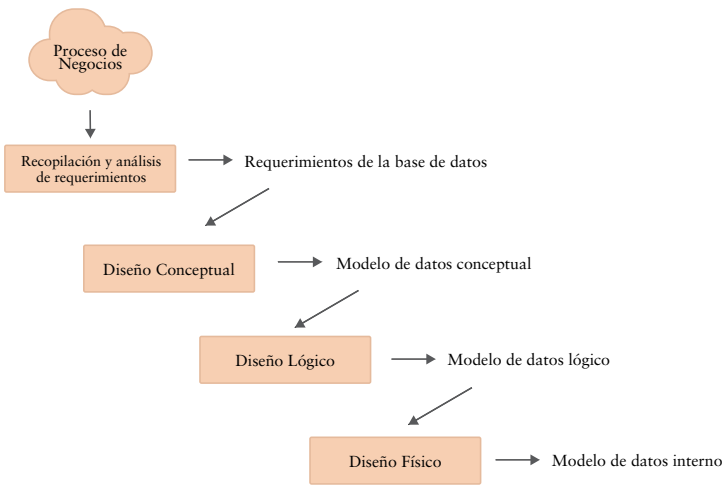
### *4.1.3 Modelado, creación y gestión de la BBDD*

En esta sección veremos los fundamentos del modelado de las bases de datos. La importancia de un buen modelamiento nace de la necesidad de mantener un buen estándar que pueda cumplir con todos los requerimientos del sistema al cual esté asociada la base de datos y que esta no necesite ser cambiada a medio camino, a menos de que nuevos requerimientos aparezcan.

Diseñar una base de datos es un proceso que contiene muchas etapas, como podemos ver en la figura 4.7. Comienza por un proceso de un negocio que necesita ser atendido, luego viene la etapa de análisis y toma de requerimientos que son posteriormente usados para construir un modelo conceptual que formalice los requerimientos y pueda ser entendido por los dueños del negocio. Una vez aprobado el modelo conceptual, comienza la creación del modelo lógico, etapa en la que ya se ha decidido el tipo de tipo de DBMS a utilizar (jerárquico, redes, relacional, etcétera). Finalmente, se realiza un modelo físico. En este, la DBMS a utilizar ya está definida y las definiciones son guardadas en el catálogo.

En las siguientes partes veremos estrategias para abordar las etapas de diseño conceptual y diseño lógico.

Figura 4.7: Proceso de diseño de una base de datos



#### *4.1.3.1 Modelado entidad-relación*

Este tipo de modelado es uno de los más populares para crear un modelo de datos conceptual amigable con los usuarios. Contiene muchos componentes útiles para crear sistemas complejos. Sus principales componentes, que veremos en profundidad más adelante, son los tipos de entidades, atributos y relaciones. Finalmente, revisaremos las limitaciones de este modelo y por qué necesitamos más herramientas para seguir con el modelado de la base de datos.

Comenzando con los componentes de un modelo entidad-relación, existen muchas variaciones y elementos que pueden ser agregados al modelo.

Un **tipo de entidad** representa un concepto del negocio con un significado no ambiguo (Lemahieu, vanden Broucke & Baesens, 2018), es decir, una representación de una categoría de entidades. Algunos ejemplos de un tipo de entidad pueden ser persona, evento o trabajador, donde una persona en específico es una entidad. En la figura 4.8 se modela una entidad cliente.

Figura 4.8: Entidad cliente

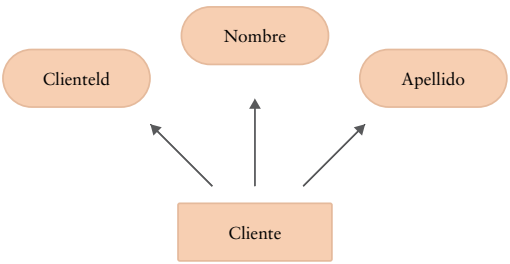
Una vez definidas las entidades, estas deben tener elementos en común, siendo estos los **tipos de atributos**, que representan propiedades de las entidades. Cada entidad debería tener un valor por cada tipo de atributo. Por ejemplo, para un tipo de entidad Cliente, sus tipos de atributos debieran ser nombre y apellido.

El **dominio** de un atributo es el rango de valores que puede llegar a tener un atributo en específico. Por ejemplo, para el sexo de una persona, su dominio será Femenino o Masculino. Es importante considerar que un dominio también puede considerar valores nulos si la situación lo requiere.

Existen múltiples categorizaciones de los atributos, los cuales se describen a continuación:

- **Atributo primario:** un atributo primario es aquel que identifica cada entidad de un tipo de entidad. Los atributos primarios son únicos para cada entidad, pero distintos tipos de entidad pueden tener valores iguales en sus llaves primarias. Este atributo puede ser una combinación de atributos. En la figura 4.9 se muestra un ejemplo de un tipo de entidad cliente, que puede identificarse con un ID debido a que múltiples clientes podrían tener el mismo nombre y apellido. Así nos aseguramos de que cada entidad pueda ser reconocida por un atributo único.

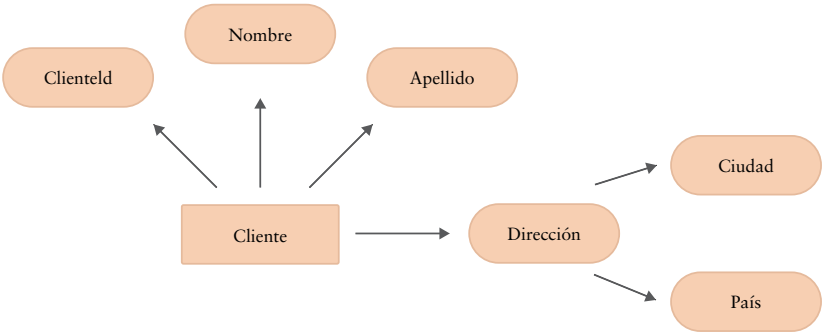
Figura 4.9: Entidad cliente con atributo primario ClienteId




En general, es buena práctica tener un tipo de atributo llamado ID para cada tipo de entidad, de manera que cada entidad se pueda reconocer de la misma manera y solo con un atributo.

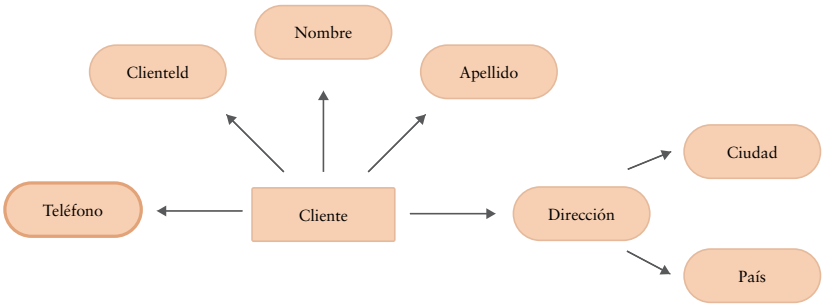
- **Atributo compuesto:** por lo general, un atributo tiene solamente un valor, pero existen atributos que se pueden dividir en varios atributos, como el ejemplo de la figura 4.10, donde un cliente tiene una dirección, la cual podría ser dividida en calle, número, ciudad, país y código postal.

Figura 4.10: Entidad cliente con atributo compuesto dirección



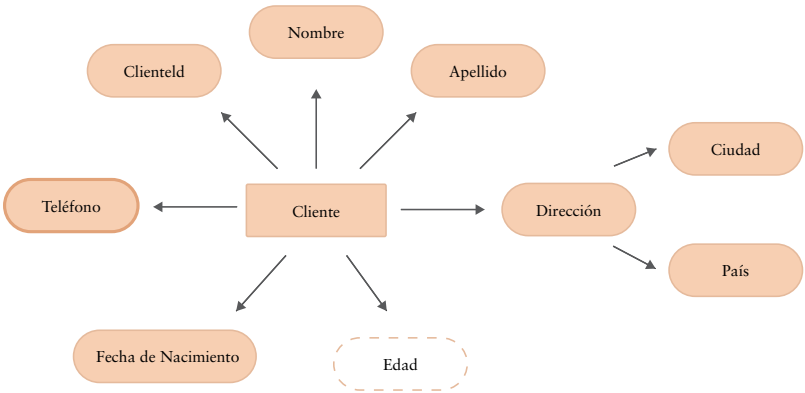

- **Atributo multivariado:** es aquel que tiene un número no definido de valores. Por ejemplo, un cliente puede tener varios números de teléfonos, como se muestra en la figura 4.11.

Figura 4.11: Entidad cliente con atributo multivariado teléfono



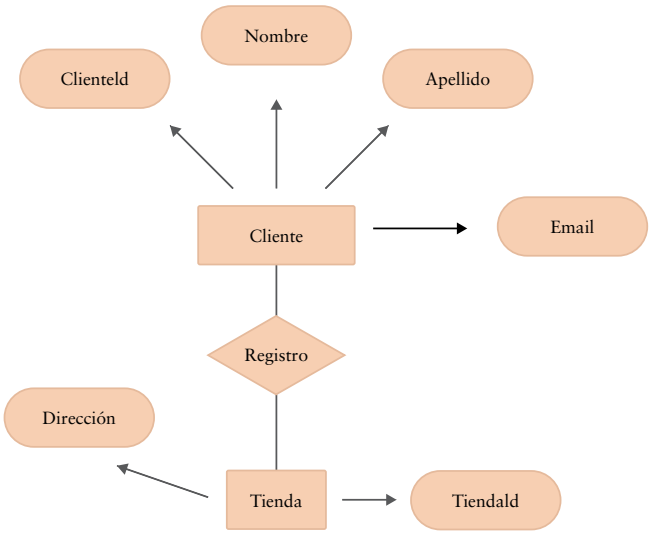

- **Atributo derivado:** un atributo derivado es un tipo de atributo en el cual su valor puede ser obtenido a partir de otro atributo. En el ejemplo de la figura 4.12 se muestra un cliente que tiene los tipos de atributos fecha de nacimiento y edad. Este último es un atributo derivado, ya que su valor puede ser obtenido de la fecha de nacimiento.

Figura 4.12: Entidad cliente con atributo derivado «edad»




Una relación es la forma de unir dos tipos de entidades. Cada relación tiene una definición que entrega un sentido a como se relacionan dos entidades. En la figura 4.13 podemos ver cómo se relaciona un cliente con una tienda. El cliente se registra en una tienda y la tienda tiene un registro del cliente.

Figura 4.13: Entidades cliente y tienda relacionadas por medio de un registro



Una relación considera un *grado*, el cual nos indica si esta puede ser entre uno o más tipos de entidades. Las relaciones unarias son aquellas que pertenecen a un solo tipo de entidad. Anteriormente vimos relaciones binarias, donde una relación une solamente dos entidades. En las figuras 4.14 y 4.15 se muestran relaciones unarias y ternarias, que unen una y tres entidades, respectivamente.

Figura 4.14: Relación unaria donde los empleados supervisan y son supervisados por empleados

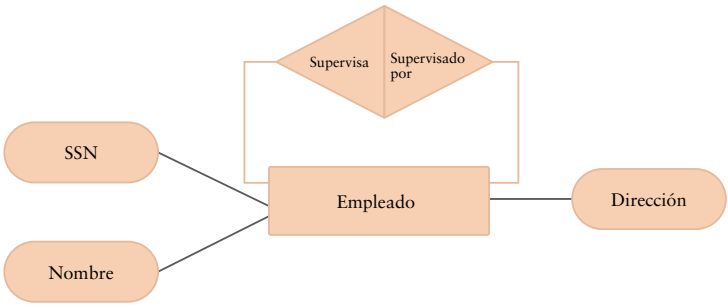
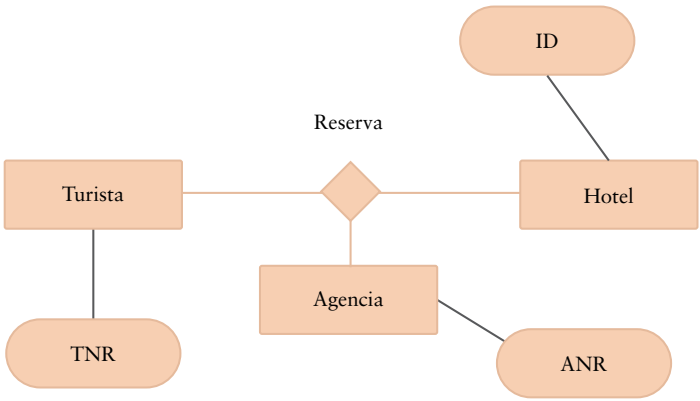



Figura 4.15: Relación ternaria donde se asocia un turista, un hotel y una agencia por medio de una reserva




En la figura 4.14 podemos ver un ejemplo de relación unaria en la cual empleados supervisan y son supervisados por otros empleados. Mientras que en la figura 4.15 vemos una relación ternaria en la cual hoteles, turistas y agencias de viaje se relacionan por medio de las reservas.

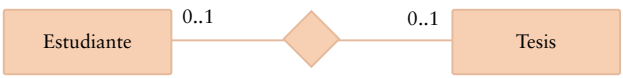
La **cardinalidad** es la cantidad mínima y máxima de entidades que puede ser partícipe de una entidad en específico (Lemahieu, vanden Broucke & Baesens, 2018). La mínima cardinalidad de una relación puede ser 0 o 1, implicando que una entidad puede existir sin estar necesariamente creada una entidad del otro tipo de entidad al que está conectada. En el caso de que la cardinalidad mínima sea 1, para que una entidad exista debe haber, por lo menos, una entidad del tipo de entidad al que está conectado. Esto último puede catalogarse como dependencia de existencia, debido a que todas las entidades necesitan participar de una relación.

La máxima cardinalidad que puede tener una relación es 1 o N. Si es 1, significa que puede haber máximo una instancia de ese tipo de entidad en específico, por ejemplo, un auto solamente puede tener un dueño. Si la cardinalidad máxima es N, significa que puede haber ilimitadas entidades de un tipo de entidad en una relación, por ejemplo, un curso puede tener muchos estudiantes.

Ahora que sabemos que las relaciones tienen máximos y mínimos en sus cardinalidades, podemos entender cómo estas funcionan en conjunto. Para ejemplificar las entidades utilizaremos a X e Y como tipos de entidades. Existen tres tipos de relaciones:

- **Uno a uno:** una relación en la cual cada entidad de X está relacionada con solo una entidad de Y. Por ejemplo, en la figura 4.16, un estudiante de una carrera de pregrado solamente puede tener una tesis. A la vez, una tesis solamente puede pertenecer a un alumno.

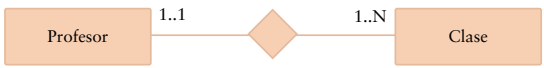
Figura 4.16: Relación uno a uno donde un alumno solo tiene una tesis




- **Uno a muchos:** en esta relación, una entidad de X puede estar asociada con muchas entidades de Y. A su vez, las entidades de Y solamente pueden estar asociadas a una entidad de X.

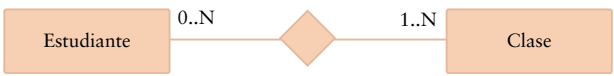
Un ejemplo de esto, en la figura 4.17, son los profesores y las clases: un profesor puede dictar muchas clases, pero cada clase es dictada solamente por un profesor.

Figura 4.17: Relación una a muchos donde un profesor puede tener múltiples clases



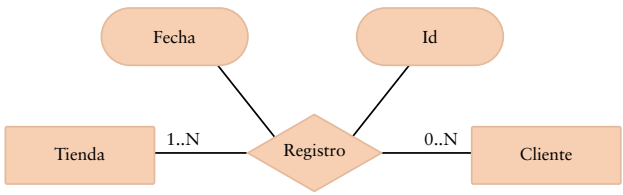

- **Muchos a muchos:** en este tipo de relación, cada entidad de X se puede asociar con muchas entidades de Y, y cada entidad de Y puede ser asociada con muchas entidades de X. Un ejemplo de esto puede ser estudiantes y cursos que se muestra en la figura 4.18, donde un estudiante puede estar asociado a muchos cursos y los cursos pueden tener muchos estudiantes.

Figura 4.18: Relación muchos a muchos donde estudiantes pueden tener muchas clases




Hay veces en que las mismas relaciones pueden tener tipos de atributos. En el caso de relaciones del tipo 1 a 1 o 1 a N, estos atributos pueden pasar a ser parte de una de las entidades participantes, no así en una relación N a M, en donde el atributo debe ser parte de la relación. En el ejemplo de la figura 4.19, un registro mantiene la información de la fecha y de su ID.

Figura 4.19: Tipos de atributos en una relación



### *4.1.3.2 Modelo relacional*

El modelo relacional es un modelo de datos formal, con una fundamentación matemática basada en la teoría de conjuntos y la lógica de primer orden. Este no tiene una representación gráfica definida, lo que hace que pueda ser usado como modelo conceptual (Lemahieu, vanden Broucke & Baesens, 2018).

Si bien no existe una representación definida del modelo relacional, este tiene componentes claves con los cuales se pueden modelar los datos. En general, el modelo relacional está compuesto por **relaciones** que contienen **tuplas**. Cada relación tiene un tipo de atributo y las tuplas contienen los atributos. A continuación, definimos cada componente en detalle.

#### *4.1.3.2.1 Componentes*

- **Relación:** es un conjunto de tuplas las cuales representan valores de una entidad real, tal como un producto o una persona. Se representa físicamente como una *tabla,* un arreglo bidimensional que contiene cada tupla y sus atributos. Corresponde al tipo de entidad del modelo entidad-relación.
- **Tupla:** es una lista de atributos que representan una entidad u objeto en una relación. Cada atributo se relaciona con un valor de cada tipo de atributo definido en la relación. Corresponde a la entidad en el modelo entidad-relación.
- **Tipos de atributos:** son los campos pertenecientes a una relación, los cuales describen sus propiedades. Se conocen como los nombres de las columnas en una tabla del modelo relacional.
- **Atributos:** son los valores pertenecientes a las tuplas, que corresponden a cada nombre de columna del modelo relacional. En una tabla los podemos reconocer como cada una de sus celdas.

*4.1.3.2.2 Tipos de llaves*

Las llaves son importantes en un modelo relacional debido a que nos permiten reconocer cada tupla dentro de una tabla y modelar las relaciones del modelo entidad-relación, es decir, cómo interactúan las entidades entre sí.

- **Llave primaria:** como se dijo anteriormente, cada entidad debe tener un atributo primario, el cual se utiliza para reconocer cada entidad. En el modelo relacional, este atributo primario es conocido como llave primaria, la cual permite que todas las tuplas tengan un valor único por el cual puedan ser reconocidas. Esta llave puede ser un atributo único o un subconjunto de atributos dentro de una tupla.
- **Llave foránea:** tal como en el modelo entidad-relación existen las relaciones como tal, en el modelo relacional estas se expresan a través de las llaves foráneas. Una llave foránea corresponde a la llave primaria de una tabla con la cual se tiene relación.

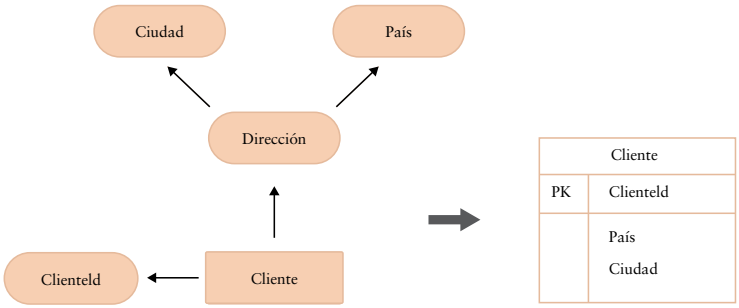
*4.1.3.2.3 Pasando de modelo entidad-relación a modelo relacional*

Pasar de un modelo entidad-relación a uno relacional no es difícil, solamente hay que seguir algunas guías como las que se explican a continuación.

Los tipos de entidades se representarán como relaciones o tablas, en donde el nombre del tipo de entidad se mantendrá en la tabla. Los tipos de atributos se representan generalmente como columnas, a menos de que estos sean compuestos, multivariados o derivados. En ese caso se tratarán de manera diferente:

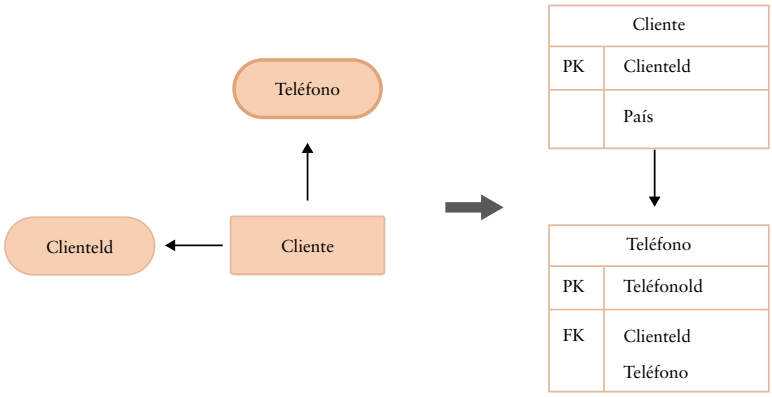
- **Atributos compuestos:** se eliminará el atributo compuesto principal y se dejarán como tipos de atributos simples aquellos que componían al atributo compuesto, como se muestra en la figura 4.20, donde una dirección se divide en ciudad y país.

Figura 4.20: Atributos compuestos modelados en un modelo relacional




- **Atributos multivariados:** este caso es más complicado ya que una relación no permite múltiples valores dentro de un atributo. Aquí se creará otra tabla que contenga el valor del atributo y una llave foránea a la tabla a la que pertenece el atributo, para cada uno de ellos. En el caso de la figura 4.21, el atributo multivariado teléfono se queda en una tabla aparte, referenciando al cliente por medio de su ID.

Figura 4.21: Atributos multivariados modelados en un modelo relacional

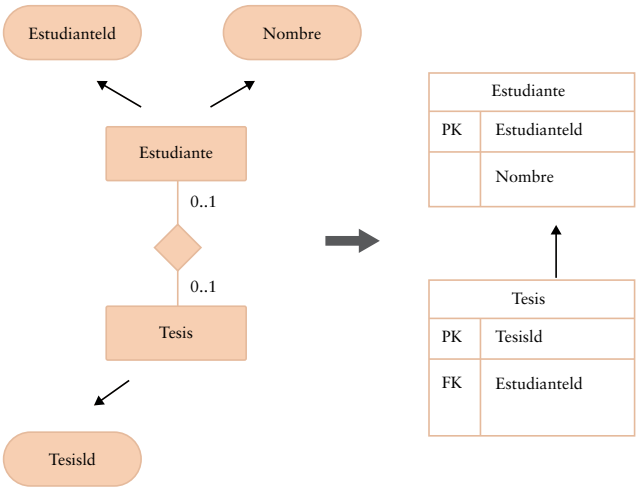



- **Atributos derivados:** estos se eliminarán debido a que podemos interpretarlos como redundancia de datos, dado que su valor se puede obtener a partir de otro atributo.

Adicionalmente, debemos tratar con las cardinalidades de las relaciones del modelo entidad-relación, fijándonos en los tres casos posibles.

- **Uno a uno:** siendo el caso más fácil, una de las dos tablas contendrá una llave foránea a la otra, lo que indicará que esas dos tablas tienen una relación entre sí. Si la relación contiene atributos, estos pasarán a ser parte de una de las tablas. En la figura 4.22, una relación entre un estudiante y su tesis se divide en dos tablas, en donde la tesis hace referencia al estudiante por medio de su ID.

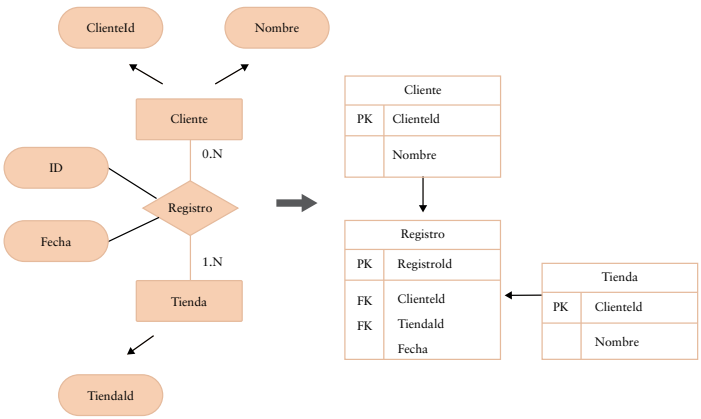
Figura 4.22: Relaciones uno a uno modeladas en un modelo relacional




- **Uno a muchos:** al igual que la relación uno a uno, una de las tablas contendrá una llave foránea con la otra tabla. A diferencia del caso anterior, el atributo que posee las entidades múltiples es el que deberá tener la llave foránea dado que así se hace la referencia desde muchas tuplas de una tabla hacia una de otra tabla. Si la relación contiene atributos, estos pasarán a ser parte de la tabla que tiene la clave foránea. Un ejemplo son los atributos multivaluados, como vimos anteriormente.

- **Muchos a muchos:** este caso es el más complicado. Es necesario crear una tercera tabla que represente la relación. Esta tabla contendrá dos llaves foráneas, una para cada tabla, y también todos los atributos que pueda haber tenido la relación en el modelo entidad-relación. Esta estrategia también sirve para cuando contamos con una relación en la que participen tres o más tipos de entidades: la tabla que represente la relación contendrá llaves foráneas para todos los tipos de entidades. En la figura 4.23 se muestra un ejemplo, donde se crea una tabla a partir de la relación registro, la cual contiene su ID, la fecha y las referencias para cliente y tienda.

Figura 4.23: Relación muchos a muchos modeladas en un modelo relacional




### *4.1.3.3 SQL*

Los sistemas de bases de datos poseen un DML, el cual nos permite operar sobre el sistema. El más popular es el lenguaje estructurado de consultas o SQL, por sus siglas en inglés. Este es un lenguaje declarativo, que puede obtener y manipular múltiples registros de un sistema de base de datos a la vez. Además, puede ser usado directamente desde alguna interfaz de un DBMS o ejecutado por un programa escrito en cualquier lenguaje (Python, Java u otros).

Para cada atributo, los DBMS tienen un tipo de variable definido, que podemos ver en la siguiente tabla.

Tabla 4.1: Posibles atributos de un DBMS

| | |
|---|---|
| *CHAR*(n) | Un texto con largo fijo n |
| *VARCHAR*(n) | Un texto con largo variable, cuyo valor máximo puede ser n |
| *SMALLINT* | Un número entero entre -32.768 y 32.767 |
| *INT* | Un entero entre -2.147.483.648 y 2.147.483.647 |
| *FLOAT* (n, d) | Un número decimal pequeño con n valores antes de la coma y d valores después de la coma |
| *DOUBLE* (n, d) | Un número decimal grande con n valores antes de la coma y d valores después de la coma |
| *DATE* | Una fecha en formato AAAA-MM-DD |
| *DATETIME* | Una fecha y hora en formato AAAA-MM-DD HH:MI: SS |
| *TIME* | Un tiempo en formato HH:MI: SS |
| *BOOLEAN* | Un valor booleano *(True* o *False)* |
| *BLOB* | Un objeto binario como una imagen, video o audio |

### *4.1.3.3.1 Creación de una base de datos*

A continuación, veremos los comandos con los cuales podemos crear, actualizar o borrar datos y tablas en una base de datos. Para esta sección utilizaremos una base de datos creada por Oracle, para bases de datos MySQL pero transportada a Postgresql. Puedes acceder a la base de datos mediante este *link:* devrimgunduz/pagila: PostgreSQL Sample Database (github.com)

- *CREATE TABLE:* comando que nos permite crear una tabla.

```
CREATE TABLE nombre_table (
nombre_columna tipo_dato restricciones,
[…]
)
```

· Columnas:

Tabla 4.2: Comandos para la creación de atributos en una tabla

| | |
|---|---|
| *NOT NULL* \| *NULL* | Define si una columna puede o no ser nula |
| *DEFAULT* valor | Define un valor por defecto a la columna si este no es especificado al momento de insertar datos |
| *UNIQUE* \| *PRIMARY KEY* | *UNIQUE* declara que todos los valores en esa columna deben ser únicos, *PRIMARY KEY* define a la columna como la llave primaria de la tabla |
| *CHECK* expresión | Se asegura de que el valor ingresado cumpla con la expresión |

· Referencia: se define la referencia como una restricción de una columna.
***REFERENCES*** nombre_tabla (nombre_columna)

· Ejemplo: creamos la tabla *Customer.*

```
CREATE TABLE customer (
customer_id integer NOT NULL,
store_id smallint NOT NULL,
first_name text NOT NULL,
last_name text NOT NULL,
email text,
address_id smallint NOT NULL,
create_date date DEFAULT ('now'::text)::date NOT NULL,
last_update timestamp with time zone DEFAULT now(), active
integer);
```

• *DROP TABLE:* este comando es para borrar una tabla.
***DROP TABLE*** nombre_tabla

· Ejemplo: *DROP TABLE customer.*

• *ALTER:* sirve para modificar una tabla o una columna.
***ALTER TABLE*** nombre_tabla
***ALTER TABLE*** nombre_tabla *ALTER* nombre_columna

· Ejemplo: añadimos la columna *activebool* a la tabla *customer.*
*ALTER TABLE customer*
*ADD activebool boolean DEFAULT true NOT NULL*,

Una vez que tenemos creadas las tablas y hemos definido las relaciones entre estas, necesitamos comandos para poder ingresar datos, editarlos y borrarlos si es necesario:

• *INSERT INTO:* se utiliza para insertar valores a una tabla. Puede ser usado para insertar uno o más valores.

· ***INSERT INTO*** nombre_columna (col1, col2, col3, …)
***VALUES*** (val1, val2, val3, …)
Ejemplo: Insertamos valores en la tabla *customer.*
INSERT INTO customer VALUES (1, 1, 'MARY', 'SMITH', 'MARY.SMITH@sakilacustomer.org', 5, true, '2020-02-14', '2020-02-15 09:57:20+00', 1);
INSERT INTO customer VALUES (2, 1, 'PATRICIA', 'JOHNSON', 'PATRICIA.JOHNSON@sakilacustomer.org', 6, true, '2020-02-14', '2020-02-15 09:57:20+00', 1);
INSERT INTO customer VALUES (3, 1, SARA, 'WILLIAMS', 'LINDA.WILLIAMS@sakilacustomer.org', 7, true, '2020-02-14', '2020-02-15 09:57:20+00', 1);
INSERT INTO customer VALUES (4, 2, 'BARBARA', 'JONES', 'BARBARA.JONES@sakilacustomer.org', 8, true, '2020-02-14', '2020-02-15 09:57:20+00', 1);

• *UPDATE:* permite actualizar los valores de tuplas que cumplan con una condición en específico.
***UPDATE*** nombre_tabla *SET* col1=val1 [, …]
[***WHERE*** condición]

· Ejemplo: cambiamos el nombre de la entidad con llave primaria 3 de SARA a LINDA.
UPDATE customer SET first_name='LINDA'
WHERE customer.customer_id=3

• *DELETE:* permite eliminar tuplas que cumplan con una condición en específico.

***DELETE FROM*** nombre_tabla
[***WHERE*** condición]

*4.1.3.3.2 Consultas*

Las consultas son declaraciones que nos permiten leer los datos de una manera específica, manifestando condiciones, orden, agrupaciones y otras funciones de agregación. Las clasificamos en tres categorías: simple, medio y avanzado.

*4.1.3.3.2.1 Simples*

Las consultas simples son aquellas que utilizan las declaraciones fundamentales. A continuación, se explican estas declaraciones y su sintaxis.

• *SELECT:* utilizado para seleccionar columnas dentro de una o más tablas. Es fundamental para todas las consultas. A continuación, vemos su sintaxis completa.

***SELECT*** [***ALL*** | ***DISTINCT***] {* | lista_columnas}
[***FROM*** lista_tablas]
[***WHERE*** condición]
[***GROUP BY*** expresión]
[***HAVING*** condición]
[***ORDER*** BY expresión]Se puede definir un alias para las columnas o tablas seleccionadas.
***SELECT*** columna1 ***AS*** col1 ***FROM*** tabla1

· Ejemplo: seleccionamos el nombre y apellido de un cliente.

**SELECT** customer_id, first_name, last_name **FROM** customer

| Customer_id | first_name | last_name |
|---|---|---|
| 1 « | MARY» | «SMITH» |
| 2 | «PATRICIA» | «JOHNSON» |
| 3 | «LINDA» | «WILLIAMS» |
| 4 | «BARBARA» | «JONES» |

• *WHERE:* permite agregar condiciones a una selección, referidas a las tablas involucradas en la selección. Estas condiciones pueden ser resultados y comparaciones de operadores sobre columnas, el valor de una columna, comparaciones entre columnas o comparaciones de columnas con valores.

· Ejemplo: seleccionamos el nombre y apellido de los clientes con *active* igual a 0.

**SELECT** customer_id, first_name, last_name **FROM** customer
**WHERE** active=0

| Customer_id | first_name | last_name | active |
|---|---|---|---|
| 16 | «SANDRA» | «MARTIN» | 0 |
| 64 | «JUDITH» | «COX» | 0 |
| 124 | «SHEILA» | «WELLS» | 0 |
| 169 | «ERICA» | «MATTHEWS» | 0 |

• *AND* y *OR:* se utilizan para dar más de una condición a *WHERE*. En el caso de *OR*, este se utiliza para agregar una condición opcional a la anterior. *AND* se utiliza para agregar una condición adicional a la anterior. Ambos operadores se basan en la lógica booleana y pueden ser agrupados con paréntesis.

c1 ***OR*** c2 ***AND*** c3 => (c1 ***OR*** c2) ***AND*** c3

· Ejemplo: seleccionamos nombre, apellido, ID de tienda y activo de los clientes que tengan *store*_id igual a 2 y *active* igual a 1.

**SELECwT** first_name, last_name, store_id, active **FROM** customer
**WHERE** store_id=2 **AND** active=0

| first_name | last_name | store_id | active |
|---|---|---|---|
| «SANDRA» | «MARTIN» | 2 | 0 |
| «JUDITH» | «COX» | 2 | 0 |
| «ERICA» | «MATTHEWS» | 2 | 0 |
| «HEIDI» | «LARSON» | 2 | 0 |
| «KENNETH» | «GOODEN» | 2 | 0 |

• Selección con reunión: cuando queremos asociar dos tablas, *WHERE* nos puede ayudar a unirlas por medio de alguna columna que tengan en común. Normalmente se usa entre tablas que tengan una llave foránea, es decir, una relación entre ellas.

· Ejemplo: seleccionamos el nombre y apellido del cliente y el ID y dirección de la tienda.

**SELECT** c.first_name, c.last_name, s.store_id, s.address_id
**FROM** customer c, store s
**WHERE** c.store_id = s.store_id

| first_name | last_name | store_id | address_id |
|---|---|---|---|
| «MARY» | «SMITH» | 1 | 1 |
| «PATRICIA» | «JOHNSON» | 1 | 1 |
| «LINDA» | «WILLIAMS» | 1 | 1 |
| «BARBARA» | «JONES» | 2 | 2 |
| «ELIZABETH» | «BROWN» | 1 | 1 |

*4.1.3.3.2.2 Medias*

• *ORDER BY:* como, en una consulta, los datos no vienen ordenados, *ORDER BY* se utiliza para ordenar el resultado de la consulta conforme a una o más condiciones. El orden en el que se especifiquen las expresiones será el orden en el que se ordenarán los datos.

*ORDER BY* col1 [*ASC* | *DESC*], […]

· Ejemplo: seleccionamos ID, nombre y apellido de clientes y los ordenamos por nombre de manera descendente.

**SELECT** customer_id, first_name, last_name **FROM** customer
**ORDER BY** first_name **DESC**

| Customer_id | first_name | last_name |
|---|---|---|
| 479 | «ZACHARY» | «HITE» |
| 174 | «YVONNE» | «WATKINS» |
| 190 | «YOLANDA» | «WEAVER» |
| 212 | «WILMA» | «RICHARDS» |
| 219 | «WILLIE» | «HOWELL» |

• *LIMIT:* utilizado para limitar la cantidad de resultados que entrega la consulta.

*LIMIT* cantidad

· Ejemplo: seleccionamos ID, nombre y apellido de los clientes pero limitamos a tres resultados.

**SELECT** customer_id, first_name, last_name **FROM** customer
**LIMIT** 3

| Customer_id | first_name | last_name |
|---|---|---|
| 1 | «MARY» | «SMITH» |
| 2 | «PATRICIA» | «JOHNSON» |
| 3 | «LINDA» | «WILLIAMS» |

• *OFFSET:* se usa para declarar las filas que queremos saltarnos antes de comenzar a mostrar datos. Normalmente se utiliza con *limit,* puesto que de esa manera podemos saber qué valores no se están mostrando.

· Ejemplo: seleccionamos ID, nombre y apellido de clientes, pero hacemos *offset* y *limit* para que nos muestre resultados desde la tercera tupla en adelante.

**SELECT** customer_id, first_name, last_name **FROM** customer
**LIMIT** 3
**OFFSET** 2

| Customer_id | first_name | last_name |
|---|---|---|
| 3 | «LINDA» | «WILLIAMS» |
| 4 | «BARBARA» | «JONES» |
| 5 | «ELIZABETH» | «BROWN» |

• PATRONES: con ellos, es posible determinar si un atributo cumple o no un patrón en específico. Se usa junto con *LIKE* para operar las columnas. Comúnmente, el signo % se usa para representar cero, uno o múltiples caracteres. El signo representa un solo carácter.

· Ejemplo: seleccionamos ID, nombre y apellido de todos los clientes que tengan dos letras «O» consecutivas en su apellido.

**SELECT** customer_id, first_name, last_name **FROM** customer
**WHERE** last_name **LIKE** '%OO%'

| Customer_id | first_name | last_name |
|---|---|---|
| 9 | «MARGARET» | «MOORE» |
| 56 | «GLORIA» | «COOK» |
| 62 | «JOAN» | «COOPER» |
| 73 | «BEVERLY» | «BROOKS» |
| 78 | «LORI» | «WOOD» |

• *DISTINCT:* por defecto, una consulta no elimina las tuplas duplicadas. Con *DISTINCT* podemos hacer que estas se eliminen explícitamente de la consulta.

• *GROUP BY:* permite agrupar los resultados de una consulta bajo uno o más atributos. Las columnas por las cuales se quiere agrupar deben aparecer en el *SELECT*. Adicionalmente, se pueden utilizar funciones de agregación, las cuales operan sobre los conjuntos agrupados. Estas columnas usadas en las funciones de agregación no deben aparecer en el *SELECT*. A continuación, listamos las más comunes:

- *COUNT:* entrega la cantidad de tuplas agrupadas bajo esa condición.
- *MIN:* entrega el mínimo valor de las tuplas agrupadas en la columna especificada.
- *MAX:* entrega el máximo valor de las tuplas agrupadas en la columna especificada.
- *SUM:* suma todos los valores de las tuplas agrupadas según la columna especificada.
- *AVG:* da el valor promedio del campo especificado.
  Los atributos que se muestran deben ser declarados en la agrupación.
- Ejemplo: contar la cantidad de clientes que tiene cada tienda según su ID.

```
SELECT store_id, COUNT(store_id) AS cantidad_de_clientes
FROM customer
GROUP BY store_id
```

| store_id | cantidad_de_clientes |
|---|---|
| 1 | 326 |
| 2 | 273 |

• *HAVING:* debido a que no es posible declarar condiciones de datos agrupados, *HAVING* nos permite aplicar estas condiciones luego de que la agrupación ha sido realizada.

· Ejemplo: contamos la cantidad de clientes por tienda y seleccionamos las tiendas que tengan más de 300 clientes.

```
SELECT store_id, COUNT(store_id) AS cantidad_de_clientes
FROM customer
GROUP BY store_id
HAVING COUNT(store_id) > 300
store_id    cantidad_de_clientes
1   326
```

*4.1.3.3.2.3 Avanzadas*

• *JOIN:* anteriormente habíamos hablado de que con *WHERE* se puede hacer una reunión entre dos tablas. Este tiene distintos tipos:

· *INNER JOIN:* o reunión interna, nos permite hacer una combinación cruzada de dos tablas, permaneciendo solo aquellas tuplas que cumplan la condición.

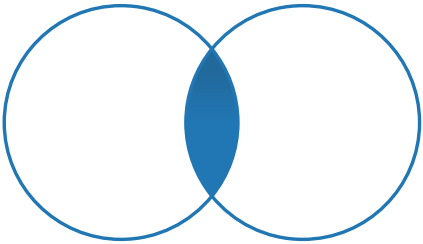
*SELECT * FROM* T1 *INNER JOIN* T2 *ON* condición

· Ejemplo: juntemos las tablas tienda y cliente según el ID de la tienda.

```
SELECT c.first_name, c.last_name, s.store_id, s.address_id
FROM customer c INNER JOIN store s
ON c.store_id = s.store_id
```

| first_name | last_name | store_id | address_id |
|---|---|---|---|
| «MARY» | «SMITH» | 1 | 1 |
| «PATRICIA» | «JOHNSON» | 1 | 1 |
| «LINDA» | «WILLIAMS» | 1 | 1 |
| «BARBARA» | «JONES» | 2 | 2 |
| «ELIZABETH» | «BROWN» | 1 | 1 |

Figura 4.24: Esquema de un *INNER JOIN*



INNER JOIN

- *NATURAL JOIN:* o reunión natural, actúa como el *INNER JOIN*, pero la columna por la cual se hace la condición solamente aparece una vez para cada tupla.

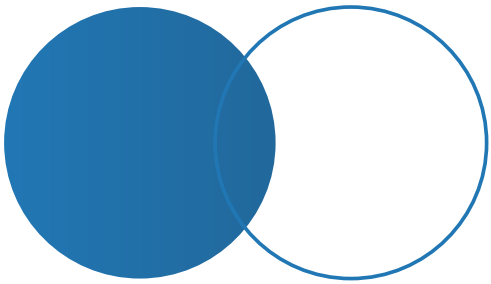
Así como las declaraciones anteriores solo nos muestran los valores que tienen en común dos tablas, se puede obligar a una reunión cuyo resultado contenga todos los valores de alguna de las tablas o de ambas. En el caso de que la tabla que no está siendo obligada a mostrar todos sus resultados no tenga ninguna coincidencia, se agregarán tuplas con valores nulos.

- *LEFT JOIN:* obliga a la tabla declarada a la izquierda a mostrar todos sus valores. La tabla declarada a la derecha se rellenará con valores nulos en caso de ser necesario.
- Ejemplo: mostremos el ID de una película, el título e ID del inventario de la unión de las tablas película e inventario, ordenado por título.

```
SELECT film.film_id,title, inventory_id
FROM film
LEFT JOIN inventory
ON inventory.film_id = film.film_id
ORDER BY title;
```

| Film_id | title | inventory_id |
|---|---|---|
| 32 | «APOCALYPSE FLAMINGOS» | 155 |
| 32 | «APOCALYPSE FLAMINGOS» | 154 |
| 33 | «APOLLO TEEN» | null |
| 34 | «ARABIA DOGMA» | 159 |

Figura 4.25: Esquema de un *LEFT JOIN*

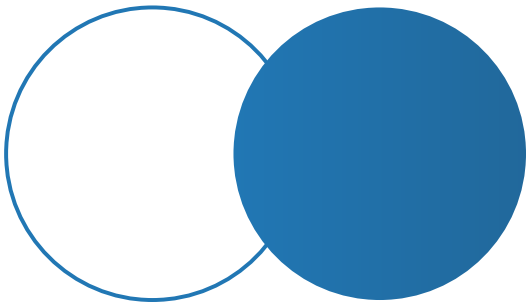


LEFT OUTER JOIN

En el ejemplo podemos ver que *inventory*_id tiene valor nulo para la fila con *film*_id 33, producto de que no existía un registro en inventario para esa película.

- *RIGHT OUTER JOIN:* se obliga a la tabla declarada a la derecha a mostrar todos sus valores, haciendo que la tabla a la izquierda rellene con valores nulos de ser necesario.
  Su funcionamiento es parecido al *LEFT JOIN*, la diferencia es que los valores que pueden ser nulos son los de la tabla izquierda.

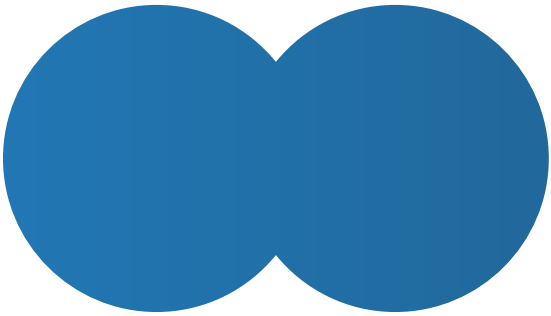
Figura 4.26: Esquema de un *RIGHT JOIN*



RIGHT OUTER JOIN

- *FULL OUTER JOIN*: ambas tablas muestran todos sus valores y se rellenan con valores nulos para aquellas tuplas de la tabla contraria con la cual no tengan correspondencia.

Figura 4.27: Esquema de un *FULL JOIN*



FULL OUTER JOIN

### *4.1.4 Bases de datos NoSQL*

Debido a la gran cantidad de datos que hoy se generan, las bases de datos relacionales no dan abasto para contener la magnitud y la forma de grandes cantidades de información. Estas no permiten cambios o nuevos datos con distinta estructura en un sistema ya diseñado y creado, pues no cumpliría con las condiciones y limitaciones predefinidas. Por esto nacen las bases de datos NoSQL, que no tratan los datos en un formato de tablas, sino que definen sus propias estructuras según el tipo de base de datos, lo que permite que puedan ser altamente escalables según necesidad.

Las bases de datos NoSQL buscan hacer el almacenamiento lo más horizontalmente escalable posible, mediante la distribución de los datos en varios *clusters* para mantener un rendimiento aceptable (Lemahieu, vanden Broucke & Baesens, 2018). Esto último provoca que el sistema no pueda asegurar la consistencia de los datos; podría generarse consistencia entre los datos con el tiempo, pero no es seguro.

En esta sección veremos los sistemas de bases de datos NoSQL más típicos: llave-valor, documentos, orientados a columnas y orientados a grafos.

#### *4.1.4.1 Llave-valor*

Como su nombre lo dice, la información es guardada de tal manera que cada registro tiene su propia llave. Estas llaves deben ser únicas para cada elemento, así podemos reconocer y acceder a cada valor en específico.

La llave especificada para cada valor no es guardada como tal en la base de datos. Existe una función llamada función de *hash* (Lemahieu, vanden Broucke & Baesens, 2018). De esta manera, la estructura de los datos es más eficiente.

Naturalmente, una sola máquina tendrá un límite de datos que pueda almacenar. Para solucionar esto, introducimos un elemento clave de las bases de datos NoSQL: el escalamiento horizontal. Debido a la naturaleza de las funciones de *hash,* para ellas es fácil crear un índice para cada valor, de manera que este puede ser reconocido en distintas ubicaciones. De esta manera, se genera un sistema distribuido de tablas de *hash*.

#### *4.1.4.2 Documentos*

Un documento es un conjunto de atributos que forman un elemento. Generalmente estos se expresan según el estándar JSON, el cual puede incluir números, valores booleanos, textos, arreglos, objetos y valores nulos como sus tipos básicos. Para un documento, cada uno de sus atributos se encuentra con una etiqueta, de manera que estos datos son semiestructurados: no todos los documentos de una base de datos deben seguir la misma estructura ni tener las mismas etiquetas, pero cada documento tiene su estructura propia. Por ejemplo, una persona puede ser representada como un documento en formato JSON:

```
{
        «nombre»: «Juan»,
        «apellido»: «Perez»,
        «sexo»: «Masculino,
        «edad»: 32,
        nacionalidad: «Chile»,
        «dirección»: {
```

```
                «calle»: «Cristóbal colón»
                «Comuna»: «comuna de mentira»
        }
}
```

Debido a que una base de datos centrada en documentos puede almacenar muchos datos, estos necesitan ser reconocidos de alguna manera, por esto, la mayoría de los sistemas permiten incluir un valor dentro del documento, único para cada elemento. Por ejemplo, MongoDB utiliza un atributo _id para reconocer cada elemento. Este atributo es autogenerado y servirá para determinar dónde se guardará el elemento.

Como los documentos tienen datos semiestructurados, de los cuales se asume que aquellos que pertenezcan a un mismo dominio tendrán estructuras parecidas, la mayoría de las herramientas permiten realizar operaciones como consultas y filtros sobre los datos. Aun así, debido a que estas bases de datos están construidas con foco en la escalabilidad, estas operaciones pueden ser bastante lentas, incluso no se relacionan elementos.

### *4.1.4.3 Orientado a columna*

Se enfocan en guardar los datos como secciones de columnas en vez de filas en donde cada una tiene la información de su entidad. De esta manera, se puede saber rápidamente qué entidades cumplen ciertas condiciones consultadas a la base de datos.

En el caso de una base de datos relacional orientada a columnas, para obtener aquellas que cumplan una condición deberíamos recorrer todas las filas, filtrando cuáles cumplen la condición y cuáles no. Esto es ineficiente desde la mirada de bases de datos orientadas a columnas, debido a que el tiempo de ejecución de la consulta dependerá de la cantidad de datos que tengamos (en el caso de que sean muchos, la consulta tomará mucho tiempo).

En el siguiente ejemplo vemos una base de datos orientada a filas y una base de datos orientada a columnas.

Tabla 4.3: Base de datos orientada a filas

| ID | Género | Título | Precio | Precio Audiolibro |
|---|---|---|---|---|
| 1 | Fantasía | Mi primer libro | 20 | 30 |
| 2 | Educación | Guía para principiantes | 10 | Null |
| 3 | Educación | SQL, El regreso | 40 | Null |
| 4 | Fantasía | El imperio del SQL | 10 | Null |

En una base de datos orientada a columnas, los valores son agrupados en una celda según la fila cumpla, o no, una condición. La base de datos anterior, con orientación a columnas se vería del siguiente modo.

Tabla 4.4: Base de datos orientada a columnas

| Género | Fantasía: 1,4 | Educación: 2,3 | | |
|---|---|---|---|---|
| Título | Mi primer...: 1 | Guía para...: 2 | SQL, el...: 3 | El imperio...: 4 |
| Precio | 20:1 | 10:2,4 | 40:3 | |
| Precio Audiolibro | 30:1 | | | |

Con esto, operaciones como «Encuentre todos los registros que tengan precio 10» pueden ser inmediatas, consultando directamente la celda de interés. Es importante notar que, al mostrar solo los datos que existen, los valores nulos ya no utilizan espacio en la base de datos.

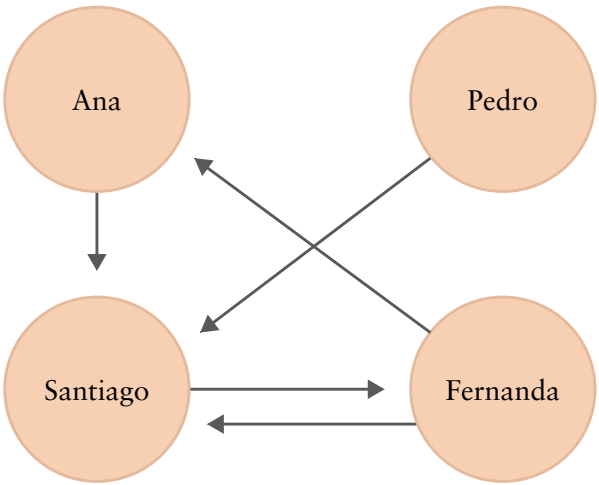
Pese a estas ventajas, el modelo también presenta desventajas. Primeramente, la operación de obtener todos los valores de una entidad sería más complicada debido a que estos valores están dispersos en distintas categorías. De la misma forma, las operaciones de reunión de tablas se verían afectadas en su rendimiento.

#### *4.1.4.4 Orientado a grafos*

Un grafo es un conjunto de nodos o vértices y aristas o líneas que conectan estos nodos. Esta unión puede ser unidireccional o bidireccional, dependiendo de lo que se quiera modelar. En el mundo de las bases de datos NoSQL, este tipo de estructura ha tomado fuerza debido a que sirve para modelar redes sociales, en las cuales existen

muchas relaciones entre distintas entidades que las componen. Por ejemplo, en la figura 4.28, tenemos una relación que se podría ver en una red social.

Figura 4.28: Relaciones entre personas modeladas como una base de datos orientada a grafos




Este tipo de relaciones se centra en modelar relaciones más que entidades, por lo que podemos modelar tipos de relaciones con cardinalidad uno a uno, muchos a uno o muchos a muchos, de manera sencilla.

Los modelos orientados a grafos utilizan un lenguaje especial llamado *cypher query language,* que permite manipular la base de datos tal como SQL, pero con una sintaxis distinta. Este lenguaje fue introducido por Neo4j, una de las bases de datos orientadas a grafos más populares.

## Referencias

Lemahieu, W., vanden Broucke, S. & Baesens, B. (2018). *Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data*. Cambridge: Cambridge University Press.

# 5. *Big Data Analytics*

Con el paso de los años, los datos se han vuelto muy importantes para las empresas, puesto que proveen de información para afrontar decisiones de envergadura o predecir conductas utilizando modelos de inteligencia artificial. Como su nombre lo sugiere, el término Big Data refiere a grandes volúmenes de datos.

Para entender la magnitud de esta tendencia, veamos el siguiente ejemplo: en 2020 Netflix estimó que sus usuarios en Estados Unidos utilizaban su plataforma un promedio de 3.2 horas al día. Considerando que en ese tiempo habían 63.7 millones de usuarios al día, hace un total de 302.8 millones de horas de reproducción al día o 6.1 mil millones de reproducciones al mes, lo que puede traducirse en 18.3 millones de TB de datos por mes[11].

Enfrentar el manejo de esta información es el principal desafío del Big Data. En esta sección revisaremos las llamadas V's del Big Data. Además, veremos tecnologías comunes utilizadas en el procesamiento de grandes cantidades de datos como Hadoop, SQL en Hadoop y Apache Spark.

## 5.1 Las 8 V´s del Big Data

El caso de Netflix es un ejemplo de la cantidad de datos que se pueden llegar a generar en un tiempo determinado, generalmente corto. Otras empresas presentan casos parecidos, como Twitter, con sus tweets y sus fotos, o Google, con la cantidad de búsquedas realizadas diariamente. También casos como Siri o Alexa, que usan los datos generados diariamente para realizar una mejora continua de sus servicios.

[11] https://www.killthecablebill.com/tv/stream/netflix/during-quarantine/

Si bien el término Big Data es bastante amplio, suele ser utilizado para referir a grandes conjuntos de datos pertenecientes a algún dominio en específico, con sus herramientas correspondientes. En el tiempo ha sido importante destacar características claves y comunes a todos los sistemas que pueden ser clasificados como Big Data. Por esto, nace en 2001 una definición del alcance de este concepto bajo el nombre de las 3 V's:

- *Volumen,* refiere a la cantidad de datos que hay en juego.
- *Velocidad,*indica que los datos se generan y recolectan con gran rapidez.
- *Variedad,* alude a la diversidad de los datos pertenecientes a dominios completamente distintos que pueden estar disponibles para una tarea en particular.

El término Big Data ha evolucionado con el paso de los años. Con ello, nuevas V's se han creado para describir características asociadas a los datos de los datos. Estas nuevas V's nacen del conocimiento y la experiencia obtenidos por la industria con el paso del tiempo y el manejo de grandes cantidades de datos:

- **Veracidad,** describe cuán fiables son los datos, considerando su consistencia y completitud (este punto es importante, ya que indica que mayor cantidad de datos no implica mejores datos).
- **Valor**, refiere a que el fin último de las operaciones con datos es definir cómo estos aportan conocimiento que sirva para generar ventajas comparativas.
- **Viscosidad**, capacidad de los datos de ser lo suficientemente resistentes para convertirse en información y ser adaptados.
- **Visualización**, grandes cantidades de datos no pueden ser visualizados por medio de gráficos tradicionales; no podríamos hacer nada con un gráfico con millones de puntos.
- **Viralidad**, nos dice qué tan rápido se dispersa la información entre Internet y la sociedad.

## 5.2 Hadoop

Hadoop es una de las herramientas más utilizadas en el mundo del Big Data. Es un *software* de código abierto utilizado para almacenamiento y procesamiento distribuido. La gran diferencia entre realizar estos procesos en un sistema común y Hadoop es que este último permite utilizar un conjunto de computadores para realizar operaciones más rápidamente. Uno de los objetivos de utilizar varios computadores es que se asume el riesgo de que alguno podría fallar, entonces, al tener varios funcionando para operar la base de datos, Hadoop puede prescindir de ese computador y funcionar con aquellos que siguen operando.

Al ser de código abierto, muchas empresas ofrecen sus propios sistemas basados en Hadoop, lo que hace fácil su uso sin tener que preocuparse por tener que crear una red de computadores propia para realizar operaciones en bases de datos. Algunas de estas empresas son Amazon, IBM, Microsoft y Oracle.

Hadoop está conformado por múltiples componentes, los cuales tienen distintas funcionalidades. A continuación, veremos cada uno y cómo podemos utilizarlos para manejar datos.

### *5.2.1 Sistema de Archivos Distribuidos*

Llamado HDFS por sus siglas en inglés, es un sistema de archivos basado en Java, utilizado para guardar datos en múltiples máquinas. También permite a los usuarios disponer de estos datos como si estuvieran accediendo a un computador, de manera fácil (tal como si estuvieran accediendo a una base datos común).

Este sistema funciona por medio de un *cluster* de nodos, entre los cuales podemos reconocer: *NameNode,* que contiene un registro sobre los datos a través de todos los nodos, como el nombre de los archivos, su tamaño y que nodo lo contiene; *DataNodes,* que manejan las operaciones de lectura y escritura sobre los datos, estos obedecen a las instrucciones generadas por el *NameNode* como las de creación y eliminación de datos.

Es en esta parte donde se puede definir el nivel de replicación para los datos a fin de considerar la tolerancia a fallo de los sistemas. En el caso de que un nodo que contenga datos falle, estos datos serán igualmente accesibles, ya que estarán almacenados en otro nodo.

### *5.2.2 MapReduce*

Habiendo visto el sistema de almacenamiento de datos, lo siguiente es su procesamiento. MapReduce es un paradigma de programación hecho por Google e implementado por Hadoop que permite procesar conjuntos de datos de manera paralela (Lemahieu, vanden Broucke & Baesens, 2018).

Su nombre viene de las operaciones *Map* y *Reduce,* las cuales son comunes a varios lenguajes de programación. Su funcionamiento está basado en aplicar funciones a múltiples elementos al mismo tiempo en vez de ir uno por uno, aumentando el rendimiento de las operaciones.

Este sistema provee un marco de trabajo de programación por el cual se pueden hacer operaciones en MapReduce directamente, pero requiere conocimiento profundo de estos sistemas por parte del programador o analista. Más adelante veremos cómo podemos utilizar SQL para operar sobre bases de datos utilizando este sistema.

#### *5.2.2.1 Yet Another Resource Negotiator (YARN)*

Si bien MapReduce realiza las operaciones de procesamiento en Hadoop, este no se encarga de coordinar los nodos ni administrar la disponibilidad de estos para realizar operaciones. La tarea de YARN es ejecutar procesos como MapReduce sobre los nodos del *cluster.* Este administra los tiempos de ejecución de las tareas sobre los computadores y define colas de procesos para optimizar su tiempo de ejecución.

La arquitectura de YARN es compleja pero necesaria para el correcto funcionamiento de Hadoop, debido a su capacidad de subdividir procesos y administrarlos. Es especialmente importante

para pasar de operaciones MapReduce hacia operaciones de SQL en Hadoop, como veremos más adelante.

## 5.3 SQL en Hadoop y *apps*

MapReduce no es la aplicación más agradable para trabajar debido a su complejidad y alta curva de aprendizaje. Por esto, se han creado aplicaciones sobre Hadoop, las cuales nos permiten utilizar Hadoop mediante SQL. Esto propicia el no tener que migrar los sistemas relacionales ya definidos con anterioridad a una arquitectura de Hadoop (que es, básicamente, un gran disco duro). A continuación, veremos Hbase, Pig y Hive, tres aplicaciones esenciales para manejar nuestro sistema de archivos utilizando tecnologías de Hadoop.

### *5.3.1 Hbase*

Corre por encima del sistema de archivos distribuido de Hadoop y puede ser considerado como uno de los sistemas NoSQL más antiguos. Más que un DBMS, es una plataforma para guardar datos debido a su falta de características como capacidades de consultas avanzadas o procedimientos almacenados (Lemahieu, vanden Broucke & Baesens, 2018).

Hbase organiza sus datos en tablas, tal como una base de datos relacional. Aun así, las diferencias son muchas: este organiza sus datos en filas, donde cada una tiene una llave y una o más columnas con valores asociados a la llave. Estas filas son ordenadas alfabéticamente por la llave, logrando que los datos que están relacionados se encuentren lo más cerca posible.

Cada columna se puede interpretar como una familia, a la cual se podría agregar una categoría o clasificación. Por ejemplo, si en una tabla de estudiantes y una columna fuera la dirección, esta podría tener una clasificación de calle y número. Veamos esto en código.

Para crear la tabla:

**create** 'estudiantes', 'nombre', 'dirección'

Debido al funcionamiento de Hbase, debemos ir ingresando los atributos de una llave uno por uno, especificando la tabla y la llave donde queremos ingresar el dato:

```
put 'estudiantes', 'juan', 'nombre', 'Juan'
put 'estudiantes', 'juan´, 'dirección:calle', 'primera calle'
put 'estudiantes', 'juan', 'direccion:numero', '234'
```

Con los comandos anteriores le dimos a la fila de la tabla, identificada con «juan», los valores de su nombre y dirección. A la dirección le asignamos dos clasificaciones, calle y número.

Si quisiéramos ver lo que acabamos de insertar:

```
get 'users', 'juan'
```

Lo que imprimiría en consola:

| COLUMN | CELL |
|---|---|
| Nombre: | timestamp:XXXXX, value=juan |
| Dirección:calle | timestamp:XXXXX, value=primera calle |
| Dirección:número | timestamp:XXXXX, value=234 |

Para cada celda, el *timestamp* sería el tiempo en formato UNIX en el cual se ingresó el dato.

También podemos eliminar valores de las columnas de los elementos

```
delete 'estudiantes, 'juan', 'dirección:numero'
```

Si quisiéramos actualizar los datos de algún registro, solo tenemos que ingresar el mismo comando que utilizamos para ingresar el dato.

```
put 'estudiantes', 'juan´, 'dirección:calle', 'segunda calle'
```

Finalmente, podemos eliminar todos los valores de las columnas asociadas a «juan»

```
deleteall 'estudiantes', 'juan'
```

Las operaciones de consultas en Hbase son bastante limitadas debido a que este sistema está pensado para ser una base de datos llave-valor solamente con operaciones de creación y eliminación. Podríamos utilizar MapReduce en Hbase, esto conllevaría a utilizar el marco de herramientas que ofrece MapReduce, lo cual puede ser

difícil de implementar. Para operaciones en Hbase, utilizaremos otras herramientas, que veremos más adelante.

### *5.3.2 Pig*

Es una plataforma de alto nivel para crear programas que corran en Hadoop. Utiliza MapReduce para ejecutar los programas que permiten a los usuarios trabajar sobre el sistema de archivos distribuido de Hadoop. Más detalladamente, podemos importar un archivo .csv y hacer consultas con la sintaxis que define Pig.

Pese a que este enfoque nos permite realizar operaciones utilizando MapReduce, se ha probado que algunas bases de datos relacionales bien estructuradas, que utilizan SQL, son más rápidas que Pig. Esta y otras razones relativas a la sintaxis de Pig hacen que este no sea tan popular dentro de los usuarios de Hadoop como otras aplicaciones.

### *5.3.3 Hive*

Una de las aplicaciones más interesantes de Hadoop, pues nos permite utilizar SQL en datos almacenados en el sistema de archivos distribuido. Estas consultas escritas en SQL son directamente traducidas por Hive a MapReduce y ejecutadas por Hadoop.

El sistema puede correr tanto en el sistema de archivos distribuido de Hadoop como en servicios como *S3* de Amazon Web Services. Esto, sumado a que utiliza SQL, es interesante por la posible integración con sistemas que ya existen en algunas empresas. La mayoría de las implementaciones aplicadas a bases de datos están hechas en SQL, lo que hace Hive muy interesante como herramienta a la cual se podría traspasar rápidamente.

Tal como Hbase, Hive está pensado para ser utilizado sobre bases de datos realmente grandes, tablas con varios millones de filas cada una. Debido a la forma en que se coordinan todos los servicios de MapReduce es que Hive puede tardar mucho en ejecutar consultas

muy sencillas, por lo que para una pequeña cantidad de datos no sirve utilizar este sistema.

## 5.4 Apache Spark

Aun cuando Hive sirve para operar con SQL en bases de datos de gran tamaño, su rendimiento no es el esperado para realizar tareas más complejas y de alto impacto, tales como extraer patrones, clasificación o limpieza de datos. Con este fin se creó Spark, un paradigma de programación centrado en conjuntos de datos distribuidos resilientes (RDD por sus siglas en inglés). Spark, al igual que las tecnologías de Hadoop, es resistente a fallas en sistemas. El objetivo de Spark es crear programas que puedan operar sobre grandes cantidades de datos, entregando la flexibilidad que podría tener cualquier lenguaje de programación como Java, Python y otros.

Spark es parecido a Hadoop ya que utiliza el HDFS como sistema de datos distribuidos y YARN para orquestar los procesos en el *cluster* de computadores. La diferencia está en que Spark busca reemplazar el procesamiento MapReduce por operaciones que puedan utilizar la memoria de manera más compartida y distribuida, lo cual permite construir más programas que solo aquellos de hacer consultas.

### *5.4.1 Spark Core*

Siendo el corazón de Spark, su *core* es quien permite realizar operaciones complejas sobre un conjunto de datos utilizando los paradigmas *map-reduce*. Hay dos características que hacen esto más rápido. La primera, es que los datos son almacenados en la memoria por el mayor tiempo posible, lo que hace que el *cluster* pueda operar sobre los datos más eficientemente. MapReduce opera sobre escrituras y lecturas en disco a través de un proceso de operaciones *map-reduce*. La segunda, es que las operaciones son divididas en subtareas asíncronas entre ellas. Estas tareas son pasadas a los trabajadores (cada computador).

La mayor ventaja de Spark Core es su facilidad y comodidad de programación. Por ejemplo, en el siguiente bloque de código, podemos cargar un archivo y contar cuántas veces aparece cada palabra y guardarlo en otro archivo.

```
#Cargamos el módulo de spark
Conf = SparkConf()
Sc=SparkContext(master=”, conf=sconf)
#Leemos el archivo
text_file = sc.textFile(«texto.txt»)
#Aplicamos funciones de Map y reduce para operar sobre el set de datos
counts = text_file.flatMap(lambda line: line.split(« «)) \
 .map(lambda word: (word, 1)) \
 .reduceByKey(lambda a, b: a + b)
#Guardamos el resultado en un archivo
counts.saveAsTextFile(«palabrasContadas.txt»)
```

Si bien, operar con MapReduce significa un gran alivio para la mayoría de las aplicaciones, aún hace falta algo que nos permita operar consultas SQL (o parecido) en nuestros datos. Spark SQL viene a resolver esto, permitiendo realizar operaciones SQL de manera directa, es decir, utilizando SQL o métodos parecidos a la librería Pandas de Python, la cual nos permite convertir los datos en un *dataframe* y utilizar métodos específicos de la librería para hacer consultas. A continuación, vemos un código de ejemplo, en el cual se realizan algunas operaciones sobre un conjunto de datos estructurados.

```
# spark es una sesión existente
df = spark.read.json(«examples/src/main/resources/people.json»)
# Mostramos el set de datos
df.show()
# +----+-------+
# | age| name|
# +----+-------+
# |null|Michael|
# | 30| Andy|
# | 19| Justin|
# +----+-------+
```

```
# Mostramos solamente la columna «name»
df.select(«name»).show()
# +-------+
# | name|
# +-------+
# |Michael|
# | Andy|
# | Justin|
# +-------+

# Seleccionamos a todos, pero aumentando su edad en uno
df.select(df['name'], df['age'] + 1).show()
# +-------+---------+
# | name|(age + 1)|
# +-------+---------+
# |Michael| null|
# | Andy| 31|
# | Justin| 20|
# +-------+---------+

# Seleccionamos a las personas con edad mayor a 21
df.filter(df['age'] > 21).show()
# +---+----+
# |age|name|
# +---+----+
# | 30|Andy|
# +---+----+

# Contamos a las personas por edad
df.groupBy(«age»).count().show()
# +----+-----+
# | age|count|
# +----+-----+
# | 19| 1|
# |null| 1|
# | 30| 1|
# +----+-----+
```

Como podemos ver, se realizaron operaciones de filtro, agrupación y lectura. Este sistema puede ser utilizado en lenguajes como Python y Java, cada uno con su librería específica.

Pese a que esto puede ser parecido a *S*QL, Spark provee una forma en el cual podemos escribir comandos en SQL, que son automáticamente transformados para ser entendidos por Spark. A continuación, podemos ver un ejemplo.

```
# Registramos el Dataframe como una vista temporal.
df.createOrReplaceTempView(«people»)
sqlDF = spark.sql(«SELECT * FROM people»)
sqlDF.show()
# +----+-------+
# | age| name|
# +----+-------+
# |null|Michael|
# | 30 | Andy|
# | 19 | Justin|
# +----+-------+
```

### *5.4.2 MLlib, Spark Streaming y GraphX*

Ya habiendo pasado por las dos funcionalidades más importantes de Spark, es necesario mencionar a MLib, Spark Streaming y GraphX.

MLib es una librería que utiliza los sistemas *RDD* de Spark para correr modelos de *machine learning*. A diferencia de Spark, Hadoop no permitía implementar este tipo de algoritmos por su naturaleza MapReduce. Esta librería ofrece modelos de regresión, clasificación y sistemas de recomendación.

Spark *Streaming* es una plataforma que utiliza Spark *Core* para proveer *analytics* en tiempo real. Provee métodos para hacer un flujo de datos a alta velocidad hacia servicios de *analytics*. Utiliza todas las tecnologías de procesamiento de Spark para realizar operaciones y las transmite a donde se le indique una vez que los datos estén listos.

GraphX es un componente que permite bajar estructuras basadas en grafos, nuevamente usando la abstracción *RDD*. Utiliza tecnologías para realizar operaciones sobre estructuras basadas en grafos y hacer los análisis sobre estas estructuras de manera más simple.

## 5.5 Herramientas de Big Data para la empresa

Si bien hemos hablado de varias tecnologías que pueden ser usadas para cargar, procesar y almacenar datos de manera eficiente y rápida, debemos aterrizar estos conceptos a herramientas que son

utilizadas hoy por las empresas. Las tres principales empresas que proveen servicios en la nube son Amazon Web Services, Microsoft y Google, cada una con sus distintas funcionalidades y ecosistemas que pueden adaptarse a nuestras necesidades.

Siendo el número uno en el mercado, con casi un 47% del mercado, Amazon Web Services se ha posicionado dentro de las empresas como una herramienta fundamental para llevar un servicio en la nube. Tiene servicios de almacenamiento de datos y *data warehousing* como Amazon RDS, Amazon DynamoDB y Amazon Redshift, procesamiento de datos con Amazon Lambda, Amazon EMR y Amazon Step, servicios de inteligencia artificial como Amazon Sagemaker y servidores dedicados como Amazon Lightsail y Amazon EC2.

Esta compañía nos provee de más de 160 servicios distintos que cubren gran parte de las necesidades que una empresa común (y no tan común) podría tener. Cada servicio contempla también la escalabilidad del sistema —podemos tener pequeños sistemas para pequeñas empresas o sistemas sumamente complejos para grandes empresas—, con mucha integración entre todos los servicios necesarios para llevar a cabo las tareas propuestas.

Otro servicio importante para considerar es Google Cloud. Este servicio también provee herramientas de *cloud computing*. Destaca por sus bajos costos de utilización de productos debido a que, por lo general, solamente pagas lo que usas, especialmente con sus servicios de computadores *Compute Engine*. Es una buena opción si queremos pagar poco por utilizar servicios de inteligencia artificial y almacenamiento, pero puede ser un poco complicado de usar, debido a la forma en que está construido.

Por último, tenemos Microsoft Azure, que al igual que sus contrapartes, tiene servicios de *cloud computing*. Es más barato que Amazon Web Services pero tiene menos servicios disponibles. Es buena opción si queremos integrar sistemas construidos con otras herramientas de Microsoft como Office 365 y PowerBI, sin ser estos últimos exclusivos.

## Referencias

Lemahieu, W., vanden Broucke, S. & Baesens, B. (2018). *Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data*. Cambridge: Cambridge University Press.

# Aplicación 2: Asignación de créditos

Asignación de créditos *(credit scoring)* es, seguramente, la tarea predictiva de negocios más tradicional. El objetivo es identificar aquellos clientes postulantes a crédito que probablemente «caerán en *default*», es decir, que no serán capaces de cumplir sus obligaciones financieras en términos de devolver el dinero prestado (Maldonado *et al.*, 2017).

Para una institución financiera, controlar el riesgo financiero es clave. Por un lado, los esfuerzos de marketing que realizan son cada vez más agresivos, con el fin de atraer nuevos clientes. Por otro lado, es muy importante ser capaz de controlar las pérdidas financieras y cuantificarlas, con el fin de provisionar adecuadamente (Siddiqi, 2007).

La asignación de créditos automatizada se populariza como una actividad clave dentro de bancos e instituciones financieras por razones principalmente regulatorias. En los años setenta se introdujo, en Estados Unidos, la *Equal Credit Opportunity Act* (ECOA), una ley que prohíbe discriminación en el otorgamiento de créditos en base a raza, color, religión o género. Si bien esta información puede ser consultada en determinadas situaciones, el proceso de asignación de créditos debe basarse en modelos analíticos transparentes que no incluyan estos atributos. La ECOA ha derivado en diferentes acuerdos internacionales, tales como Basilea II/III, suscritos por muchos países en el mundo. La introducción de estas guías de cumplimiento ha reforzado el interés en este tipo de estrategias (Baesens *et al.*, 2016).

Si bien el uso de modelos matemáticos para la asignación de créditos es un requerimiento legal, existen varias ventajas para utilizar este tipo de estrategias automatizadas en comparación con estrategias cualitativas (Baesens *et al.*, 2016; López y Maldonado, 2019; Siddiqi, 2007):

- La automatización de decisiones acelera el proceso de otorgamiento de créditos, lo que es vital para mantenerse competitivos. Decisiones basadas en el juicio de un comité de expertos puede demorar semanas, con la posibilidad de desmotivar a clientes postulantes a crédito.
- La interpretabilidad de los modelos no solo evita discriminación, sino además permite identificar patrones importantes. Por ejemplo, la comprensión de las variables más relevantes que permiten distinguir entre «buenos» y «malos» pagadores es valiosa para tomar decisiones táctico-estratégicas sobre adquisición de nuevos clientes o políticas de retención.
- El uso de modelos matemáticos, que sean capaces de balancear de forma objetiva y óptima toda la información disponible, suele tener una mejor capacidad predictiva que decisiones cualitativas tomadas por expertos.
- Los modelos analíticos no solo son capaces de reducir costos gracias a la automatización en el proceso de otorgamiento. Además, estos modelos permiten identificar las fuentes de datos que son relevantes para el problema mediante selección de atributos (ver capítulo 2). La construcción de estos modelos no solo considera fuentes de datos internas provenientes de la aplicación de crédito, también suele incluir información externa que se adquiere de un buró de crédito *(credit bureau),* tales como Experian o Equifax. Alternativamente, las mismas entidades financieras siguen procesos costosos de recolección de información, ya sea en horas de ejecutivos o de los clientes mismos. Los modelos transparentes ofrecen herramientas cuantitativas que permiten evaluar si esta información merece ser incluida en los modelos o no, reduciendo el costo de recolección de variables.

Es importante indicar que, además del desarrollo de modelos para nuevos aspirantes a crédito *(application scoring)*, existen enfoques para evaluar la capacidad financiera de clientes existentes *(behavior scoring)*. Desde el punto de vista de modelación, los enfoques de solución suelen ser similares. Sin embargo, los modelos de *behavior scoring* permiten un análisis financiero más profundo, ya

que se puede considerar información del comportamiento de pago del cliente hasta ese momento.

## 1. Definición de variable objetivo y predictores

De acuerdo con regulaciones vigentes (Basilea II/III), un cliente «malo» se suele definir como aquel que acarrea deudas por 90 días o más, considerando una ventana de tiempo (ventana de desempeño). Esta ventana puede ser de un año, por ejemplo.

Para la construcción de la matriz de datos es importante identificar, para cada cliente que recibió un crédito, el momento exacto en que se crea la nueva cuenta. En base a este hito se define la ventana de desempeño y con esto el valor de Y. Además, la construcción de variables que conforman la matriz X debe resumir la información hasta este hito. Esto es muy importante, ya que un error común es calcular variables tales como la edad o antigüedad del cliente en base a la fecha de hoy (el día en que el analista decide construir la base de datos). Sin embargo, esta «fecha de hoy» no debe considerarse. Puede sonar confuso que a cada cliente se le considere una fecha diferente para calcular la edad, pero es lo correcto, ya que debe reflejar su edad al momento de recibir el crédito (recordar capítulo 2).

Con respecto a la disponibilidad de datos, se sugiere como regla general contar con al menos 2000 clientes «buenos» y 2000 «malos» obtenidos en una ventana de tiempo predefinida (Siddiqi, 2007). El problema de desbalance suele estar presente en esta aplicación, por lo que los «malos» pagadores suelen ser más escasos.

## 2. Inferencia de los rechazados

Un aspecto importante en riesgo de crédito es el concepto de *reject inference*. La muestra de datos para la construcción de modelos es intrínsecamente sesgada, debido a que solo se cuenta con una etiqueta (valor de Y) para los clientes *aceptados* por la entidad financiera. Los clientes que fueron *rechazados* solo cuentan con información al momento de la evaluación (valor de X).

Asumiendo que el «modelo original» que se utiliza para aceptar o rechazar clientes funciona razonablemente bien y es capaz de distinguir entre «buenos» y «malos», los datos que genera tienen un sesgo de selección. Esto ocurre porque los aceptados (la muestra) tienen, en teoría, una mayor proporción de buenos pagadores que el total de clientes que solicitan crédito (la población, aceptados y rechazados). Como consecuencia, el riesgo intrínseco de la muestra es menor que el de la población.

Existen diversos métodos para corregir este sesgo, que buscan inferir el desenlace que hubiese tenido el crédito de aquellos candidatos que fueron rechazados por la entidad. Los métodos de inferencia exitosos toman en consideración el hecho de que estos clientes son, en promedio, más riesgosos que los que sí cuentan con una etiqueta (Siddiqi, 2007; Thomas, 2009).

*Augmentation* (Banasik *et al.*, 2003) es una de las técnicas más comunes para este problema. Consiste en construir un clasificador con los clientes aceptados, para luego evaluar a los rechazados, calculando su probabilidad de ser un mal pagador *(probability of default* o *PoD*). El umbral de decisión se ajusta de tal manera que la tasa de «malos» sea mayor en el conjunto de los rechazados en comparación con la de los aceptados. Finalmente, el modelo se reestima considerando tanto aceptados como rechazados, incluyendo etiquetas reales e inferidas (Banasik *et al.*, 2003).

Volviendo a la regla general sobre disponibilidad de datos, se suele requerir 2000 clientes rechazados adicionales con el fin de realizar la inferencia (Siddiqi, 2007).

## 3. Weight of Evidence

El *weight of evidence (WOE)* es una transformación que, si bien podría estar en el capítulo 2, se utiliza casi de forma exclusiva en riesgo de crédito, junto a la regresión logística como clasificador. La idea es discretizar las variables numéricas en diferentes intervalos *(binning)*. Sin embargo, no se utilizan variables *dummies* para introducir las categorías al modelo. En vez de eso, cada intervalo recibe

un «valor representativo» en base a la relación de clientes buenos y malos en el *bin*. En otras palabras, la variable se discretiza y se vuelve a hacer numérica.

Para cada *bin*, el «valor representativo» que recibe viene dado por la siguiente fórmula:

Ecuación App2. 1

$$WOE = ln(\frac{\%Buenos}{\%Malos}) \cdot 100$$

La racionalidad de esta transformación tiene que ver con la interpretación de la regresión logística. La regla de decisión de este clasificador se basa en la suma ponderada de las variables. Sin embargo, las multiplicaciones se pueden evitar al transformar todas las variables en categóricas. En este caso, se puede expresar la regla de decisión en forma de sumas, donde cada intervalo contribuye con un valor. De esta forma es mucho más simple explicar las razones de aceptación o rechazo, tanto al cliente como a la entidad que supervisa a la entidad financiera.

Existe, además, una ventaja adicional. Es importante recordar que el origen de las *scorecards* se remonta a finales de los setenta. En esos tiempos, los cálculos de los puntajes de los clientes eran hechos por los ejecutivos de forma manual. El cálculo de una *scorecard* usando papel y lápiz es mucho más simple que estimar la regresión logística en su forma tradicional, con multiplicaciones y exponenciales[12].

## 4. Construcción de una *Scorecard*

Una vez que todas las variables se transforman mediante *WOE,* el modelo se entrena utilizando una regresión logística para obtener los coeficientes de cada variable. Es común aplicar una transformación a la probabilidad de incumplimiento que permita llevarla a una

[12] La fuente de este argumento es una conversación informal sostenida con Lyn Thomas (q.e.d.), autor de *Thomas* (2009) y uno de los padres del riesgo de crédito.

escala más intuitiva (por ejemplo, de 1 a 1000). Cada rango de un atributo tendrá asociado un *score* positivo o negativo (por ejemplo, si la edad del aplicante está entre 20 y 30, sumar 10 puntos). El *score* final será la suma de los *scores* obtenidos en cada variable.

Una estrategia de *scorecard* muy común es la logarítmica, que se basa en la siguiente fórmula:

Ecuación App2.2

$$Score = Escala + Factor \cdot ln(odds)$$

En donde los *odds* son las chances de ocurrencia del evento (incumplimiento crediticio). Por ejemplo, *odds* de 50 equivalen a que uno de cada 50 aplicantes será mal pagado. Los *odds* están ligados directamente a la suma ponderada que calcula la regresión logística.

Se definen los puntos para duplicar las *Odds* (PDO) como la cantidad de puntos necesaria para duplicar los *odds*. De esta forma, se obtiene el sistema de ecuaciones que permite llegar a una *scorecard* en base a la regresión logística:

Ecuación App2.3

$$\begin{gathered} Score = Escala + Factor \cdot ln(odds) \\ score + pdo = Escala + factor \cdot (2 \cdot odds) \\ ln(odds) = -(\sum_{i=1}^{n} \beta_i \cdot x_i + \beta_0) \end{gathered}$$

Los valores de la escala, *odds* y pdo suelen ser definidos por el científico de datos. Una solución común es Escala=600, *odds*=50 y pdo=20, que quiere decir que a los 600 puntos se tienen chances de 50 es a uno. Con esto:

Ecuación App2.4

$$Factor = \frac{20}{ln(2)} = 28.85$$

$$Escala = 600 - 28.85 \cdot ln(\frac{50}{1}) = 487.12$$

$$Score = 487.12 - 48.85 \cdot (\sum_{i=1}^{n} \beta_i \cdot x_i + \beta_0)$$

De esta forma tenemos nuestra transformación a un puntaje recibiendo como entrada los coeficientes de la regresión logística y el valor de cada variable. Si cada variable es binaria, entonces la presencia de ese nivel j aportará un puntaje que viene dado por $-48.85*\beta_j$, y de esta forma se convierte el problema en una serie de sumas y restas.

## Referencias


Baesens, B., Roesch, D. & Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. New Jersey: John Wiley and Sons.

Banasik, J., Crook, K. & Thomas, L. (2003). «Sample selection bias in credit scoring models». *Journal of the Operational Research Society,* 54(8), 822-832.

Thomas, L. C. (2009). *Consumer Credit Models: Pricing, Profit and Portfolios*. Oxford, Oxford Scholarship Online.

López, J. & Maldonado, S. (2019). «Profit-based Credit Scoring based on Robust Optimization and Feature Selection». *Information Sciences,* 500, 190-202.

Maldonado, S., Bravo, C., Pérez, J. & López, J. (2017). «Integrated framework for profit-based feature selection and SVM classification in credit scoring». *Decision Support Systems,* 104, 113-121.

Siddiqi, N. (2007). *Credit risk scorecards: developing and implementing intelligent credit scoring*. New Jersey: Wiley & Sons.

# 6. *Predictive Analytics* I: modelos transparentes

## 6.1 Introducción

La ciencia de los datos se basa en comprender el pasado y predecir el futuro analizando datos. En los capítulos anteriores nos enfocamos en la primera tarea mediante *descriptive* y *diagnostic analytics*. Nuestro viaje por la modelación predictiva comienza en este capítulo, con la tarea de clasificación y, en particular, la construcción de modelos transparentes para apoyar la toma de decisiones.

Clasificación y regresión son tareas del aprendizaje supervisado, en donde se utiliza el conocimiento *a priori* del comportamiento de un conjunto de observaciones, llamado *conjunto de entrenamiento*. En palabras simples, el conjunto de entrenamiento corresponde a la información que en el pasado explicó un fenómeno en particular, como por ejemplo la fuga de clientes, riesgo crediticio de un cliente, etcétera. Los modelos de aprendizaje supervisado son capaces de aprender utilizando esta información, para luego ser aplicados a datos nuevos con el fin de predecir el fenómeno en estudio.

La transparencia es una virtud importante de los modelos, sobre todo en negocios. El proceso predictivo suele venir acompañado por la toma de decisiones, por lo que comprender las reglas aprendidas por un modelo para decidir puede ser extremadamente valioso. Pensemos en el problema de predicción de fuga. Una máquina puede elegir a qué clientes ofrecer una política de retención para «mantenerlos contentos» en la empresa en caso de identificar que tienen un riesgo alto de fuga. Esto es una decisión operativa o «del día a día». Sin embargo, puede ser muy valioso comprender cuáles son los factores

que determinan el riesgo de fuga de un cliente y su ponderación en el modelo predictivo. Esto permitiría tomar decisiones de más largo plazo e importancia (decisiones táctico-estratégicas), tales como el diseño de una campaña de marketing para retención, la definición del presupuesto de campaña o incluso ajustes en el posicionamiento de la empresa (Baesens, 2014).

Este capítulo comienza con los dos modelos principales de clasificación transparentes: regresión logística y árboles de decisión. Después formalizaremos la estrategia para comparar y evaluar el desempeño predictivo de los modelos, discutiendo algunos desafíos importantes, como el desbalance de clases. En los capítulos siguientes estudiaremos modelos que carecen de interpretación, pero que pueden superar, en desempeño, a los transparentes. Estos modelos se conocen como «caja negra» (Han *et al.*, 2011).

## 6.2 Regresión logística

La regresión logística es el primer modelo predictivo que se estudiará en este libro, ya que es el que recomendamos construir primero para tener una noción de los resultados predictivos que se pueden alcanzar. Además de la transparencia, entre sus virtudes se cuentan que es un modelo bastante simple y que entrega buenos resultados sin necesidad de ajustar parámetros.

Hablaremos de parámetros como las «perillas» que podemos mover para llegar a un buen modelo. Mientras más perillas tenga un modelo, más flexible es. Sin embargo, la flexibilidad tiene un costo asociado, ya que es un proceso no solo tedioso y potencialmente costoso en tiempo, sino que puede traer errores en la modelación. En este sentido, el hecho de que la regresión logística no tenga perillas, lo hace una excelente primera opción para abordar problemas de clasificación.

## 6.3 Regresión lineal y su adaptación a clasificación

La regresión logística es un modelo lineal en términos de los coeficientes que multiplican a las variables predictivas. Dicho de otro modo, estudiaremos relaciones lineales entre las variables de la matriz X y su efecto en Y (James *et al.*, 2017).

Para entender la regresión logística, primero introduciremos el concepto de regresión lineal. Asumiendo que el fenómeno a estudiar es numérico (problema de regresión) y tenemos una variable dependiente o explicativa en la matriz X, la regresión lineal busca los ponderadores $\beta_0$ y $\beta_1$ de forma de cometer el menor error posible al explicar $y$ con $x$ de forma lineal:

Ecuación 6.1

$$y = \beta_0 + \beta_1 x$$

Es importante notar que tanto el lado izquierdo como el derecho de una regresión lineal son numéricos, es decir, están en un rango $(-\infty, +\infty)$. El desafío se tiene en clasificación, donde $y$ es nominal. Por ejemplo, $y$ puede tomar los valores «bueno» o «malo» para un problema de riesgo de crédito, indicando si un cliente del conjunto de entrenamiento fue capaz de cumplir sus obligaciones crediticias («bueno») o no («malo»). Otro ejemplo es «fuga» o «no fuga», identificando si un cliente deja la compañía o no.

¿Cómo hacemos «conversar» nuestro lado izquierdo (nominal) con nuestro lado derecho (numérico)? Lo haremos en tres etapas: la primera es simplemente pasar las etiquetas a una variable *dummy* para darle sentido matemático (recordar capítulo 2). Volviendo al ejemplo de riesgo crediticio, la convención es que «0» identifica un buen pagador y «1» a un mal pagador. En fuga de clientes, un cliente fugado suele ser etiquetado como «1».

Ahora tenemos, al menos, tanto el lado izquierdo como el derecho numérico. Sin embargo, aún no «conversan»: $\{0,1\} \neq (-\infty, +\infty)$. El segundo paso es aplicar una función que tome nuestro lado derecho $(-\infty, +\infty)$ y lo lleve a valores entre 0 y 1. Para esto, aplicaremos la

función de enlace logística o *logit*. Esta función, también llamada «sigmoidal», toma la siguiente forma:

Ecuación 6.2

$$p(z) = \frac{1}{1 + e^{-z}}$$

Ecuación 6.3

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

La variable $z$ puede tomar valores entre $(-\infty, +\infty)$ y es nuestro lado derecho de la regresión, la cual «entra» en una función exponencial de la ecuación anterior. De esta forma, se transforma el lado derecho en un número entre 0 y 1. Verifiquemos esto: si $z$ toma un valor muy grande, la expresión $e^{-z}$ será cercana a 0, lo que se traduce en un $p(z)$ cercano a 1. Por el contrario, si $z$ toma un valor muy negativo, $e^{-z}$ será cercano a $\infty$, por lo que $p(z)$ se vuelve muy cercano a cero.

Dado que $p(z)$ toma valores entre 0 y 1, se puede interpretar como la probabilidad de ocurrencia del fenómeno estudiado (el definido como «1»). En asignación de créditos, por ejemplo, «1» identifica al cliente mal pagador, por ende, $p(z)$ se interpreta como la probabilidad de incumplimiento crediticio. El siguiente ejemplo muestra el cálculo de esta probabilidad para una regresión logística con tres variables explicativas (edad, deuda e ingreso):

Ecuación 6. 4

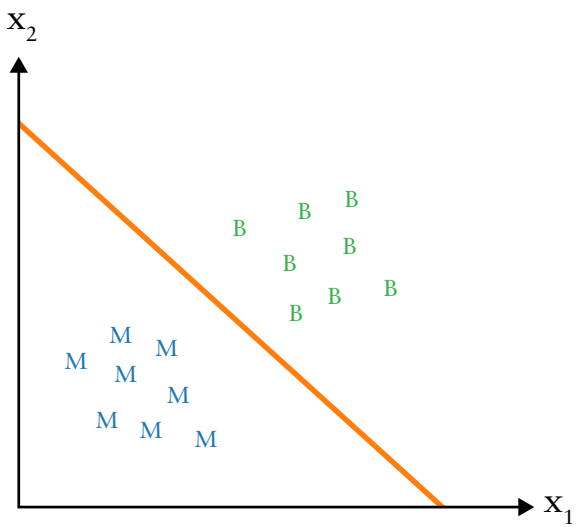
$$p(y = "malo" \mid edad, deuda, ingreso) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_{edad} + \beta_2 X_{deuda} + \beta_3 X_{ingreso})}}$$

Aún no estamos listos, ya que $\{0,1\} \neq (0,1)$, es decir, nuestro lado izquierdo tiene valores que son exactamente 0 o 1, mientras que el derecho tiene probabilidades. El tercer paso es definir un umbral, es decir, un corte para discretizar las probabilidades y convertirlas

en ceros y unos. El umbral estadístico por defecto es 0.5, es decir, si $p(z)>0.5$, la predicción es clase 1. Volviendo al ejemplo, si el cálculo de $p(z)$ es 0.7, entonces el modelo sugiere rechazar al cliente y no otorgarle crédito, ya que su probabilidad de incumplimiento es mayor a 0.5. El valor del umbral, sin embargo, no es necesariamente 0.5 siempre, depende de los beneficios y costos de clasificación. Volveremos al tema de la elección del umbral más adelante en el libro.

Para finalizar la adaptación de regresión lineal, discutiremos la interpretación geométrica de la regresión logística. El modelo se interpreta como un hiperplano separador, lo que equivale a una recta cuando consideramos dos dimensiones de la matriz X. En la igura 6.1, podemos observar una serie de clientes con una edad y un ingreso. Los clientes pueden ser buenos («B») o malos («M») en términos de la devolución del crédito. La recta separa de la mejor forma posible los dos grupos de clientes. Los coeficientes beta se estiman de tal manera que se cometa el menor error posible, y representan las pendientes del hiperplano óptimas.

Figura 6.1: Interpretación geométrica de una regresión logística



## 6.4 Interpretación y selección de atributos

Los coeficientes beta que conforman el hiperplano se pueden interpretar como la ponderación que tiene cada variable en la ecuación 6.3. A partir de estos coeficientes podemos extraer dos tipos de *insights* relevantes:

- El **signo del beta** indica la dirección hacia donde apunta el patrón. De acuerdo con la ecuación (6.3), si el signo del beta asociado a una variable es positivo, entonces un mayor (menor) valor de esa variable aumenta (disminuye) la probabilidad de ocurrencia del evento.
- La **magnitud del beta** indica, hasta cierto punto, la relevancia de la variable. De acuerdo con la ecuación (6.3), si una variable está siendo multiplicada por cero o un valor cercano, esta pierde completamente su influencia sobre la probabilidad de ocurrencia del evento y, por ende, es candidata a ser descartada del modelo.

Desarrollemos el primer punto con el ejemplo de la ecuación (6.3). En riesgo de crédito, uno esperaría que mientras mayor sea el ingreso de un cliente, el riesgo (en promedio) debería ser menor. Entonces, el **signo esperado** del beta para la variable ingreso es negativo, de acuerdo con lo que sabemos de la tarea predictiva. De la misma forma, el signo esperado del beta para la variable deuda es positivo, es decir, mientras mayor sea la deuda los clientes son, en promedio, más riesgosos.

Muchas veces ocurre que los signos esperados no coinciden con los estimados por el modelo. Por ejemplo, en proyectos que hemos realizado en asignación de créditos nos han aparecido deudas con signo negativo, es decir, la regresión dice que, a mayor deuda, menor riesgo. Esto es una completa contradicción con lo que debería pasar en la realidad.

¿Por qué ocurre esto? La mayoría de las veces no se debe a errores en las variables. Esto lo podemos descartar rápidamente en el análisis bivariado (ver capítulo anterior). El problema se debe a las correlaciones entre variables, especialmente entre variables

redundantes. La mayoría de las veces en que hemos identificado signos inconsistentes se debe a la presencia de otra variable altamente correlacionada con el predictor cuyo coeficiente tiene el signo incorrecto.

Recomendamos verificar si el signo de cada coeficiente es consistente con lo esperado de acuerdo con lo que sabemos de la aplicación. Este análisis puede evitar que implementemos modelos con errores que puedan invalidar nuestro trabajo. Dado que la máquina es incapaz de saber si el signo es consistente con lo esperado (no tiene «prejuicios» sobre los coeficientes), esta tarea es exclusiva del analista.

Vamos ahora al segundo *insight:* la magnitud del beta. Si el beta asociado a una variable es muy cercano a cero, esta variable puede ser descartada en un proceso de selección de atributos. Notar que un coeficiente nunca es exactamente cero y que no es fácil tener una noción clara de cuándo es lo suficientemente cercano a cero como para descartar la variable. La decisión depende de la desviación estándar de los coeficientes. Utilizaremos un *test* de hipótesis para evaluar la relevancia de la variable.

Como se indicó en el capítulo pasado, la selección de atributos ayuda no solo a acelerar los procesos de decisión y mejorar la capacidad predictiva de los modelos, también favorece la interpretación del modelo. Es mucho más simple explicar decisiones como el otorgamiento de créditos cuando en la ecuación del modelo se tienen cinco variables en vez de 50.

En el capítulo pasado nos enfocamos en los métodos de filtro, los cuales pueden considerarse como preprocesamiento. La selección de atributos es realizada antes de aplicar un modelo. Con la regresión logística, explicaremos el concepto de selección de atributos **envolvente** o *wrapper.* La idea general es construir varios modelos con diferentes conjuntos de variables, evaluar el desempeño de

clasificación y elegir el mejor. Dado que hacer esto mediante «fuerza bruta» y probar a ciegas muchas combinaciones es un proceso demandante en tiempo, aprovecharemos las «pistas» que nos da la regresión logística con relación a la relevancia de los coeficientes.

Formalmente, la regresión construye estimadores que siguen una distribución normal. Una opción posible de estadígrafo a construir es el estadístico $z$ (diferente al chi cuadrado estudiado en la selección de atributos por filtros). Este se obtiene dividiendo el valor del coeficiente por su desviación estándar y sigue una distribución normal unitaria.

Lo más importante es entender el concepto de p-valor, el cual se interpreta como **la probabilidad de equivocarse al rechazar la hipótesis nula**. Independiente del estadígrafo y su construcción, solo nos basta mirar este valor para concluir sobre un *test* de hipótesis. La regla más común es la siguiente: se rechaza la hipótesis nula si el p-valor es menor a 0.05.

Ahora, debemos tener claridad sobre cuál es la hipótesis nula $H_0$. En este caso evaluamos si un coeficiente en particular es estadísticamente cero, o no:

Ecuación 6.5

$$H_0: \beta_n = 0$$

Ecuación 6.6

$$H_1: \beta_n \neq 0$$

En conclusión, dependiendo de la significancia de la variable podemos evaluar si la variable es relevante. Si su p-valor es sobre 5%, la variable es candidata para eliminarse. No obstante, en el caso de existir varias variables irrelevantes, estas deben eliminarse una por una, es decir, se descarta la más irrelevante y luego se evalúa el modelo con las nuevas variables. A este método se le conoce como selección de atributos «hacia atrás» *(backward elimination)*.

El problema de utilizar este método es que, al eliminar la variable más irrelevante, la nueva estimación puede considerar variables relevantes aquellas que en primera instancia no lo eran, por lo tanto, ir eliminando una por una cada variable, obteniendo distintos

niveles de significancia en cada iteración, puede resultar una tarea engorrosa para el investigador. Para evitar esta iteración manual, existen estrategias automáticas tanto en R (método glmStepAIC del paquete CARET) como en Python (enfoque RFE del paquete SKLEARN), que descartan la o las variables irrelevantes de forma iterativa sin que el usuario deba supervisar el proceso.

Es importante notar que la eliminación hacia atrás no es la única alternativa. Existe el enfoque de selección de atributos «hacia adelante» *(forward selection)*, en donde estos se van agregando al modelo partiendo desde considerar solo una variable. Adicionalmente, también hay métodos bidireccionales que combinan la idea de ir hacia adelante o hacia atrás según necesidad (Baesens *et al.*, 2016).

En resumen, la regresión logística presenta las siguientes características:

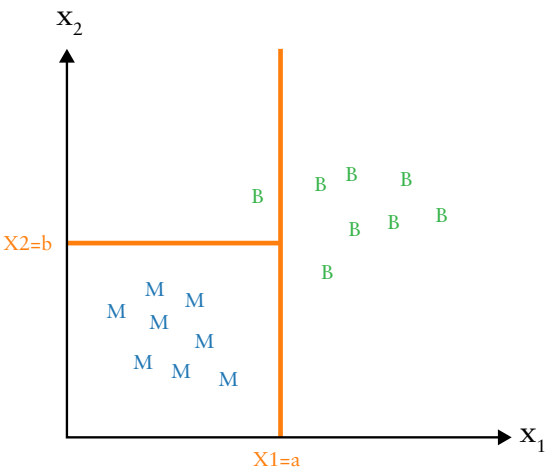
- Es un modelo simple y que suele llegar a buenos resultados predictivos «a la primera», es decir, sin pasar por procesos complejos de exploración de parámetros. Si bien no suele ser el modelo que obtiene los mejores resultados en comparación con modelos sofisticados, rara vez suele estar muy lejos del mejor desempeño posible.
- Su interpretación es una ventaja importante. Se basa en el análisis de los coeficientes y la opción de realizar selección de atributos envolventes.
- Su simpleza y falta de flexibilidad es también su principal desventaja: solo captura relaciones lineales entre las variables. Puede que otros modelos funcionen mejor cuando los patrones son más complejos.
- El modelo de regresión logística suele estimar los coeficientes mediante la estrategia de máxima verosimilitud. Esta estrategia asume que las filas son independientes entre sí. Este es un supuesto en general bastante razonable, pero puede no cumplirse.

## 6.5 Árboles de decisión

El segundo modelo predictivo que se estudiará en este libro corresponde a la familia de los árboles de decisión. Este modelo utiliza una estructura jerárquica de toma de decisiones, eligiendo la mejor variable a ramificar de forma recursiva.

Los árboles de decisión poseen una estructura completamente distinta a la regresión logística. En vez de construir un hiperplano (ver figura 6.1), esta técnica se interpreta como cortes horizontales y verticales realizados de forma sucesiva. La figura 6.2 presenta precisamente cómo sería un clasificador de árbol para los datos representados en la figura 6.1: dos variables en un plano cartesiano para el problema de separar buenos («B») y malos («M») pagadores de crédito.

Figura 6. 2: Interpretación geométrica de un árbol de decisión




Geométricamente el árbol construye cortes horizontales y verticales en las variables (en este caso X1), en el que establece ciertas reglas. Por ejemplo, en la figura 6.2 podemos apreciar que en X1=a se establece un corte vertical, el cual es capaz de separar dentro de lo posible las observaciones «B» de las «M». Por otro lado, en la variable X2 se establece un corte horizontal X2=B, el cual permite

al árbol afirmar que si X2 < b y X1 < a, la observación se clasifica como «M».

Los árboles de decisión construyen relaciones del tipo «si sucede esto, entonces pasa esto». Este algoritmo en sus principios estuvo diseñado solo para variables nominales, no obstante, hoy es adaptable a variables numéricas, definiendo un umbral para poder ramificar con ellas. De todos modos, un árbol de decisión no logra captar cabalmente relaciones numéricas del tipo «a mayor ingreso menor riesgo». Por esta razón, la regresión logística suele ser mejor opción cuando el modelo se compone de variables principalmente numéricas, mientras que el árbol suele ser mejor cuando las variables son principalmente nominales, tales como educación (básica, media, universitaria) o género (masculino o femenino), etcétera.

Una de sus principales ventajas de los árboles es la capacidad de interpretación, ya que solo observando el árbol podemos apreciar cuáles son las variables imperativas para poder discriminar una clase de otra. Por ejemplo, en la aplicación de riesgo de crédito, observando el árbol podemos notar cuáles son las variables que incidieron en la clasificación de riesgo, por lo tanto, es fácil justificar por qué un cliente es aceptado o rechazado para la obtención de un crédito.

### *6.5.1 Algoritmo y construcción del árbol de decisión*

Los árboles de decisión están construidos por diferentes algoritmos. Dentro de ellos, los más populares son: C4.5, CART, CHAID y ID3. No obstante, si bien poseen diferencias estructurales, todos los algoritmos se construyen en base a decisiones tales como:

- **Decisión de división**: responde a la pregunta ¿dónde divido a la variable? (por ejemplo: Edad>30; Ingreso<$300.000; etcétera).
- **Decisión de asignación**: responde a la pregunta ¿qué clase asigno al nodo raíz?
- **Decisión de detención**: responde a la pregunta ¿hasta dónde ramifico?

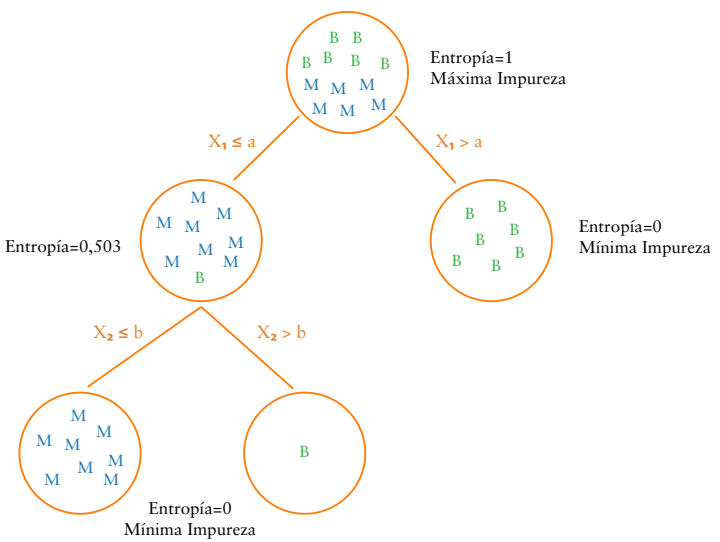
*Decisión de división y asignación:*

El árbol comienza con un nodo raíz que contiene todos los datos y posteriormente se escoge el atributo que mejor clasifica los datos. Ahí comienza la ramificación.

El desafío principal radica en encontrar el atributo que mejor clasifica. En palabras simples, corresponde al atributo cuya ramificación separa de mejor forma ambas clases. Para separar las clases de manera óptima, necesitamos tener una medida para cuantificar el nivel de *desorden* o impureza en los datos. Las medidas más populares son entropía (C4.5), Gini (CART) y análisis de Chi-cuadrado (CHAID). Para efectos de este libro se presentará la primera medida.

La entropía puede definirse como el desorden que hay en base a la proporción de buenos y malos («B» y «M»). La máxima entropía se encuentra en 1, esto quiere decir que están en igual proporción los buenos y los malos («B» y «M»). Por el contrario, la entropía 0 quiere decir que solo hay buenos o solo hay malos. La entropía será entre 0 y 1 en casos intermedios, como se aprecia en la figura 6.3. En ella se presenta el resultado del proceso de ramificación asociado a los datos de la figura 6.2 como una estructura de árbol.

Figura 6.3: Ejemplo de un árbol de decisión basado en entropía



Formalmente, la fórmula de la entropía o desorden para un problema de dos clases (seguiremos con el ejemplo de buenos y malos pagadores de crédito) está definida de la siguiente forma:

Ecuación 6.7

$$Entropía: E(S) = -p_M log_2(p_M) - p_B log_2(p_B)$$

La fórmula anterior recibe como información $p_M = 1 - p_B$, que corresponde a la proporción de buenos y malos pagadores.

Considerando el ejemplo en la figura 6.4, procederemos a explicar el algoritmo del árbol de decisión. El contexto de negocio considera un grupo de 15 clientes de una tienda de tecnología, quienes han tomado la decisión de comprar o no un dispositivo *tablet* de última generación. Se consideran estos datos como un «histórico» del cual el algoritmo del árbol extraerá aprendizaje para, en un futuro, predecir si un cliente comprará este *tablet* en estudio.

Figura 6.4: Datos de entrenamiento para compra de *tablet*

| ID | Edad | Nivel de ingreso | Universitario | Deuda pendiente | Clase: compra *tablet* |
|---|---|---|---|---|---|
| 1 | <25 | medio | no | sí | no |
| 2 | <25 | bajo | sí | no | sí |
| 3 | >40 | medio | sí | sí | sí |
| 4 | <25 | medio | sí | no | sí |
| 5 | 25-40 | medio | no | no | sí |
| 6 | 25-40 | alto | sí | sí | sí |
| 7 | >40 | medio | no | no | no |
| 8 | <25 | alto | no | sí | no |
| 9 | <25 | alto | no | no | no |
| 10 | 25-40 | alto | no | sí | sí |
| 11 | >40 | medio | no | no | no |
| 12 | >40 | bajo | sí | sí | sí |
| 13 | >40 | bajo | sí | no | no |
| 14 | 25-40 | bajo | sí | no | sí |
| 15 | <25 | medio | sí | sí | sí |

ID simplemente es un identificador de clientes que no aporta información al problema de negocio, razón por la cual no se considera para el cálculo de entropía. La columna «Edad» corresponde a la discretización de esta variable en tres niveles: menor a 25 años, entre 25 y 40 años y mayores a 40. La columna «Nivel de ingreso» corresponde a la discretización de la variable en tres niveles: alto, medio y bajo. La columna «Universitario» es una variable dicotómica que presenta el valor «sí», si la observación representa a un cliente universitario, y «no», si no lo es. «Deuda pendiente» también es una variable dicotómica que indica si el cliente posee actualmente deuda. Finalmente, podemos observar la variable «Compra tablet», la cual es nuestra variable objetivo que representa la decisión de compra de *tablet* para un cliente en particular.

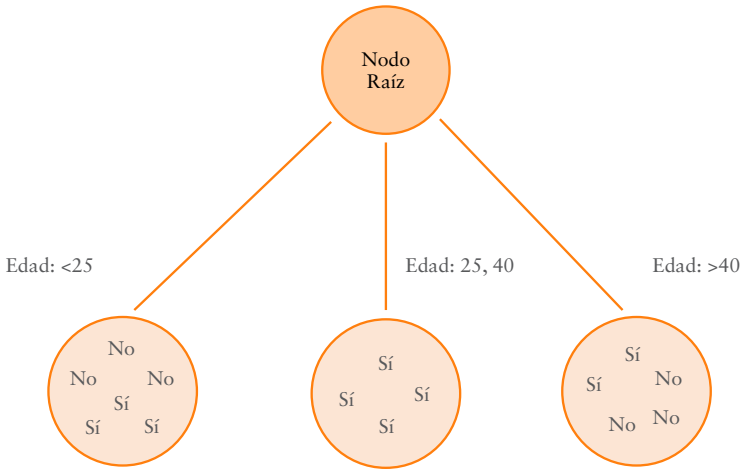
Como mencionamos, lo primero que se debe hacer es escoger el atributo que mejor clasifique. Recordemos que la ganancia de información es la suma ponderada de las entropías de cada nodo. Por lo tanto, considerando que tenemos seis personas que no compran la *tablet* y nueve que sí lo hacen, la entropía del nodo raíz es:

Ecuación 6.8

$$E_{Nodo\ raíz} = -9/15\ x\ log_2(9/15) - 6/15\ x\ log_2(6/15) = 0{,}971$$

Lo que haremos ahora es ramificar cada variable desde el nodo raíz y elegir la que logra reducir de mayor manera el desorden. Comenzaremos con el atributo edad, lo que visualmente quedaría de la siguiente forma:

Figura 6.5: Ramificación del atributo «edad»




Para los jóvenes menores de 25 años contamos cuántos clientes compran y cuántos no. Según la figura 6.5, tenemos que tres personas compran y tres no lo hacen. Repetimos el ejercicio para las personas entre 25 y 40 años, donde vemos que todos compran, por lo tanto, no existe desorden en los datos (entropía=0). Finalmente, en el tercer nodo correspondiente a personas con edad sobre 40 años, observamos que dos personas sí compran y tres no lo hacen.

Con esta información se procede a calcular la entropía (o desorden) de cada nodo ramificando con edad:

Ecuación 6.9

$$E_{(Edad<25)} = -3/6\ x\ log_2(3/6) - 2/6\ x\ log_2(2/6) = 1$$
$$E_{(Edad:\,[25,40])} = -4/4\ x\ log_2(4/4) - 0/4\ x\ log_2(0/4) = 0$$
$$E_{(Edad>40)} = -3/5\ x\ log_2(3/5) - 2/5\ x\ log_2(2/5) = 0{,}971$$

La entropía total de esta variable será la suma ponderada de las entropías de sus nodos:

Ecuación 6.10

$$E_{Edad} = (\frac{6}{15}\ x\ 1\ +\ \frac{4}{15}\ x\ 0\ +\frac{5}{15}\ x\ 0{,}971) = 0{,}724$$

Una vez calculada la entropía de cada nodo, procederemos a calcular la ganancia de información. Esta medida se entiende como una disminución ponderada de las entropías de cada nodo:

Ecuación 6.11

$$Ganancia_{Edad}\ =\ E_{Nodo\ raíz} - E_{Edad}$$

Ecuación 6.12

$$Ganancia_{Edad}\ =\ 0{,}971 - 0{,}724\ =\ 0{,}247$$

0,247 sería la ganancia si ramificamos edad. No obstante, tenemos otras tres variables candidatas a ramificar. Se procede a calcular la ganancia en reducción de entropía para ellas. Queda propuesto para el lector realizar los cálculos de forma detallada.

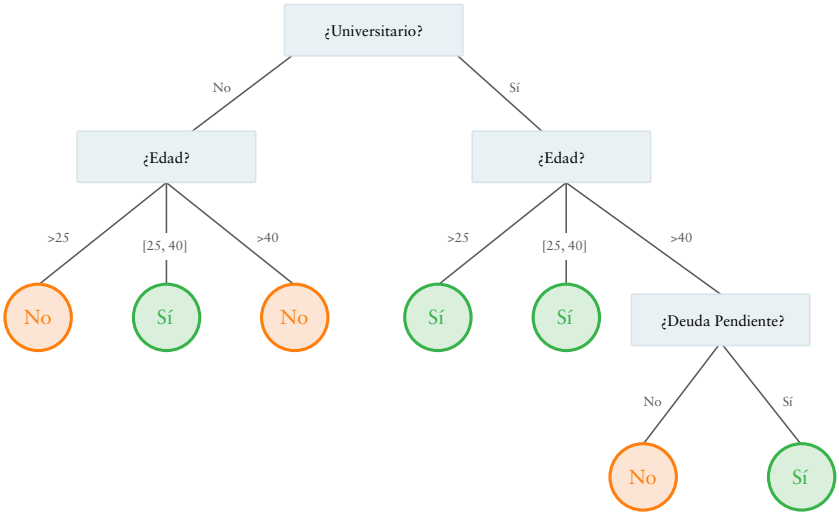
Ecuación 6.13

$$\boldsymbol{Ganancia_{Ingreso}\ =\ 0,971 - 0,943\ =\ 0,028}$$
$$\boldsymbol{Ganancia_{Universitario}\ =\ 0,971 - 0,693\ =\ 0,278}$$
$$\boldsymbol{Ganancia_{Deuda}\ =\ 0,971 - 0,936\ =\ 0,035}$$

Ahora que tenemos todas las ganancias, el atributo que escogemos para ramificar es Universitario, con una ganancia de 0,278. Por lo tanto, la primera ramificación se realiza con esta variable, que puede considerarse como la más relevante de acuerdo con este modelo.

Dado que los dos nodos que se desprenden de la ramificación Universitario no poseen entropía 0, el árbol se sigue ramificando bajo el mismo algoritmo explicado anteriormente. De esta forma, el árbol resultante es:

Figura 6.6: Árbol de decisión compra de *tablet*




Una virtud del árbol de decisión es que permite hacer selección de atributos con el mismo modelo predictivo, el cual se encarga de eliminar por sí solo las variables que no necesita. En el ejemplo anterior, existe una variable (Nivel de ingreso) que el modelo decidió ignorar y no ramificar con ella, llegando a excelentes resultados. Esto se conoce como selección de atributos empotrada o *embedded:* el mismo modelo se encarga de expulsar la información irrelevante. Notar que la regresión logística no hace esto: siempre construye un modelo utilizando toda la información que le entregamos. La regresión logística nos da una mano al momento de definir qué variable deberíamos sacar, pero lo cierto es que requiere la construcción de varios modelos para descartar todo el ruido. El árbol de decisión, sin embargo, puede hacerlo solo con un único modelo.

*Decisión de detención:*

Evidentemente, si el árbol se divide muchas veces (muchas ramas) existirán nodos con muy poca información, por lo tanto, será un árbol demasiado detallado. El árbol puede estar «aprendiendo»

del ruido intrínseco del problema y no predecir de forma adecuada. En este sentido, el cliché «menos es más» suele ser bastante cierto, al menos coherente con el principio de parsimonia conocido como la Navaja de Occam: ante dos modelos que explican la realidad de forma similar, debería elegir el más simple. Un modelo simple es más probable que sea una mejor representación del fenómeno y que extrapole mejor los patrones que encuentra en el conjunto de entrenamiento.

Si un árbol tiene una cantidad excesiva de ramas, entonces puede estar siendo más complejo de lo necesario. Esto se conoce como **sobreajuste**, ya que el modelo comienza a ajustarse al ruido en los datos. Para evitar esto, existen diferentes formas de «podar» el árbol, es decir, de limitar el crecimiento del árbol para que no ramifique demasiado. Esto se puede hacer ya sea controlando la profundidad del árbol (por ejemplo, que no ramifique más de tres veces de forma sucesiva) o bien limitando inferiormente la cantidad de ejemplos que quedan en las hojas (por ejemplo, que ninguna hoja tenga menos de 20 observaciones).

Las estrategias de poda pueden ser definidas de forma arbitraria, pero lo más común es buscar la configuración que entregue mejores resultados predictivos. Aquí es donde se complejiza el proceso. Llamaremos a estas formas de podar los **parámetros del problema**, es decir, las perillas que nos entregan flexibilidad.

La elección de los parámetros se realiza mediante un proceso de **validación**, que corresponde a evaluar el modelo en un conjunto de datos que permanecen ocultos al modelo. De esta forma se realiza una especie de «marcha blanca», es decir, vemos qué pasaría si implementamos nuestro modelo para predecir con datos donde sí conocemos las etiquetas reales. Este proceso de validación será explicado a continuación.

En resumen, el árbol de decisión presenta las siguientes características:

- Su principal virtud es la interpretabilidad, siendo capaz de explicar sus decisiones en base a una serie de reglas y considerando un conjunto reducido de variables.
- A diferencia de la regresión logística, este modelo obtiene mejores resultados si el conjunto de datos está compuesto principalmente por variables nominales.
- Requiere la calibración de uno o varios parámetros de poda que se consideren. El objetivo de este proceso es encontrar un buen balance entre simpleza y flexibilidad.

## 6.6 Validación de modelos

Hemos visto ya dos modelos predictivos y sus propiedades teóricas, sin embargo, esto no es suficiente para decidir qué modelo elegir. Nos gustaría quedarnos con el que funciona mejor para nuestros datos.

En ciencia de los datos existe el famoso teorema «*no free lunch*» *(NFL)* o «no hay almuerzo gratis», que indica básicamente que no existe un modelo mejor que otro solo en base a sus propiedades teóricas. La consecuencia de *NFL* es que los modelos deben probarse de forma empírica, ya que un método que funciona bien con un conjunto de datos puede fallar en otro.

Para validar los modelos antes expuestos, debemos conocer en primer lugar el término **particionamiento**. Este concepto hace referencia a la partición de la base de datos en un conjunto de entrenamiento y otro de testeo, de tal forma que el modelo pueda aprender con el primero y comprobar la efectividad de este aprendizaje con el segundo.

Existen diferentes estrategias de validación de modelos. En este libro estudiaremos dos: la validación simple y la validación cruzada.

### 6.6.1 *Validación simple (holdout)*

En este método se particionan los datos de manera aleatoria, considerando aproximadamente un 70% de los datos en el conjunto de entrenamiento y el 30% restante en conjunto de testeo. Por regla general, el conjunto de entrenamiento debe ser más grande que el de testeo, con al menos un 65% de los datos. Los porcentajes utilizados pueden variar de acuerdo con los criterios del científico de datos.

Es importante que las proporciones de ambas clases de la variable objetivo queden balanceadas en ambos conjuntos. Este procedimiento se conoce como *holdout* estratificado. En el caso aplicado de fuga de clientes, por ejemplo, se espera que el conjunto de entrenamiento y de testeo posean la misma proporción de clientes fugados y no fugados. De no ser así, pueden no ser tan fidedignas las medidas de desempeño de los modelos.

El problema de la selección de datos al azar se agrava cuando las clases son desbalanceadas, es decir, una clase se encuentra sobrerrepresentada con relación a otra. Esto es un problema común en aplicaciones de riesgo de crédito o fuga de clientes. El caso extremo es que tengamos tan mala suerte que no quede ningún ejemplo de la clase minoritaria en el conjunto de entrenamiento y el modelo no se pueda estimar. Volveremos al tema del desbalance de clases en la siguiente sección de este capítulo.

Para estratificar el particionamiento, lo que debemos hacer es simple: seleccionamos a todos los elementos de una de las clases y tomamos el 70% de estos para el conjunto de entrenamiento y el 30% restante para testeo. Repetimos esto para la segunda clase y combinamos ambos conjuntos de entrenamiento, por un lado, y ambos conjuntos de testeo por el otro.

### 6.6.2 *Validación cruzada (crossvalidation)*

Después de pasar semanas, o quizás meses, preprocesando datos, ¿deberíamos contentarnos con construir un modelo de cada tipo y

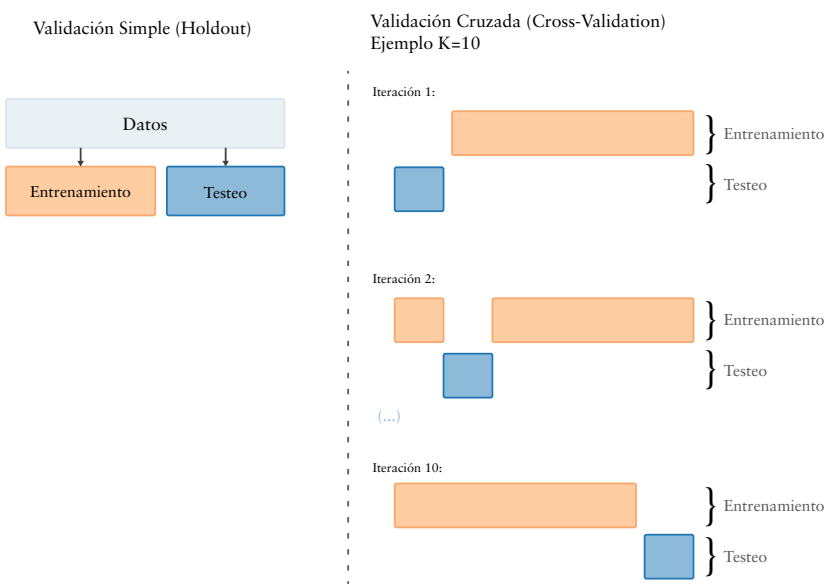
elegir el mejor? Si tenemos un conjunto de datos de tamaño mediano, podemos terminar la validación simple en menos de un minuto.

El teorema *NFL* sugiere también que los modelos tienen desempeños parecidos, mientras que la elección del mejor modelo puede definirse por decimales. Puede que incluso el mejor modelo cambie si corremos dos veces *holdout,* dependiendo de cómo se repartan los datos en entrenamiento y testeo.

Entonces, ¿cómo podemos hacernos cargo de esta aleatoriedad? Una solución más robusta es la **validación cruzada,** la cual particiona los datos en *k* conjuntos disjuntos de forma estratificada. La idea es construir *k* experimentos para cada modelo, de forma que cada subconjunto de datos es *k-1* veces entrenamiento y una vez testeo. Finalmente, el desempeño en cada conjunto de testeo se promedia.

La figura 6.7 presenta ambas estrategias y sus diferencias. Se recomienda dividir en *k=10* subconjuntos, de tal forma que se entrene con nueve de estos y se testee en el restante. De esta forma, el desempeño final será el promedio de 10 experimentos, lo que lo hace mucho más robusto que la validación simple.

Figura 6.7: Diferencia entre *holdout* y validación cruzada



Para finalizar, una recomendación que podemos hacer es combinar validación simple y validación cruzada de tal forma de tener dos procesos de evaluación de modelos. Primero, haremos una validación simple y seleccionaremos el 70% de los datos para la construcción del modelo, ocultando el 30% restante.

Si tenemos un modelo sin parámetros (como la regresión logística), lo construimos en este conjunto de entrenamiento. Si queremos calibrar parámetros, sin embargo, no deberíamos hacerlo mirando el 30% de testeo. Este conjunto debe permanecer oculto y ser utilizado exclusivamente para comparar entre clasificaciones de diferente naturaleza. Sería un error comparar modelos con diferente número de parámetros en un mismo proceso de validación, ya que terminaríamos favoreciendo a aquellos que tienen más parámetros.

Lo que sugerimos es hacer validación cruzada dentro del conjunto de entrenamiento, particionando en 10 este 70% de los datos y evaluar los parámetros (por ejemplo, diferentes criterios de poda para el árbol de decisión) con estos datos. Una vez que se elige la mejor configuración de parámetros, entrenamos el modelo con el conjunto de entrenamiento completo y lo comparamos con la regresión logística utilizando el 30% oculto. Esta estrategia se conoce como **validación anidada** *(nested validation)*.

## 6.7 Medidas de desempeño

Como hemos mencionado anteriormente, las medidas de desempeño son parte fundamental del proceso de validación, pues gracias a estas podemos cuantificar qué tan buenas son las predicciones de distintos modelos. En este apartado hablaremos de medidas de desempeño para modelos de clasificación binaria.

Para entender el desempeño de un modelo de clasificación, debemos considerar que existen cuatro posibles escenarios, los cuales pueden verse representados en la siguiente **matriz de confusión***:*

Figura 6.8: Matriz de confusión

| | | REAL | |
|---|---|---|---|
| | | Clase Positiva | Clase Negativa |
| PREDICCIÓN | Clase Positiva | Verdadero Positivo [VP] | Falso Positivo [FP] |
| | Clase Negativa | Falso Negativo [FN] | Verdadero Negativo [VN] |

Siguiendo con la aplicación de fuga de clientes, definimos la clase **fuga** como la clase positiva y **no fuga** como la clase negativa. Por lo tanto, si el modelo predice como fuga un cliente que efectivamente se fugó, estaremos ante un «Verdadero Positivo» (VP). Del mismo modo, definiremos con «Verdadero Negativo» (VN) si el modelo predice a un cliente como un no fuga, cuando en la realidad efectivamente no se fugó.

También se puede dar el caso de que el modelo no predice correctamente la clase esperada. Llamaremos «Falso Positivo» (FP) si el modelo predice como fuga cuando en realidad el cliente no lo hace y «Falso Negativo» (FN) cuando el modelo predice como no fuga a un cliente que efectivamente sí se fugó.

### *6.7.1 Acierto global*

Esta medida calcula la proporción de acierto (para ambas clases) con relación al total de casos:

Ecuación 6.14

$$Acierto\ Global\ (Accuracy) = \frac{VP + VN}{VP + FP + FN + VN}$$

*Accuracy* no diferencia entre un acierto VP o acierto VN, lo que puede llevar a contar una historia incorrecta del desempeño. Lo más común en las diferentes aplicaciones es que se valore más un tipo

de acierto por sobre el otro, pero el acierto global los considera a ambos similares.

En riesgo de crédito, por ejemplo, predecir incorrectamente un mal pagador y darle un crédito que no devolverá íntegramente suele ser más costoso que equivocarse con los buenos pagadores (Thomas, 2009). En la misma dirección, predecir incorrectamente un cliente que se fugará y no hacer algo para evitarlo es más costoso que equivocarse con los no fugados (Verbeke *et al.*, 2017). Volveremos a estos ejemplos en los capítulos de aplicación.

### 6.7.2 *Sensibilidad y especificidad*

El primer paso para hacerse cargo de las diferencias entre los costos de clases es calcular el acierto tanto para la clase positiva (sensibilidad o *sensitivity*) como negativa (especificidad o *specificity*). Para la primera medida, corresponde a la cantidad de observaciones que el modelo predice correctamente como positivas respecto al total que efectivamente lo son:

Ecuación 6.15

$$Sensibilidad = \frac{VP}{VP + FN}$$

Por otro lado, la especificidad equivale a la cantidad de observaciones que el modelo predice correctamente como negativas respecto al total que efectivamente son negativas:

Ecuación 6.16

$$Especificidad = \frac{VN}{VN + FP}$$

Volviendo al ejemplo de fuga, supongamos que el 90% de los clientes no se fugan y son leales a la compañía en un periodo

determinado. Un modelo que predice que el 100% no se fuga tendrá un acierto global de 90%, lo que puede sonar muy bien. Sin embargo, el modelo tiene un 0% de sensibilidad y especificidad de 100%, y no tiene ninguna utilidad para identificar fugados.

### *6.7.3 Medidas de acierto balanceado*

En muchas aplicaciones, como en medicina, sensibilidad y especificidad no se combinan ya que los costos no son realmente estimables. Pensemos en un modelo predictivo para el cáncer: ¿cuál es el costo de equivocarnos y no detectar cáncer cuando el paciente sí tenía, causando potencialmente su muerte?, ¿cuál es el costo de equivocarnos e identificar cáncer en un paciente sano y hacerlo pasar por un tratamiento delicado (como quimioterapia) de forma innecesaria? Claramente, estos riesgos no son combinables. Por esto la investigación en medicina se enfoca en maximizar sensibilidad y especificidad por separado.

Sin embargo, existen muchas aplicaciones donde nos gustaría resumir el desempeño del modelo en una medida única que podamos comparar, pero que sea **balanceada**, es decir, que no sea influenciable por el desbalance de clases como en el ejemplo anterior de fuga. Para ello, discutiremos en este libro las medidas *g-mean* y AUC.

La medida *g-mean* corresponde a la media geométrica entre la sensibilidad y especificidad:

Ecuación 6.17

$$Media\ Geométrica\ = \sqrt{Sensibilidad\ x\ Especificidad}$$

Esta medida es balanceada ya que requiere que tanto sensibilidad como especificidad sean altos para llegar a un número alto (cercano a 1). En el ejemplo de fuga anterior, *g-mean* resulta ser 0, ya que la sensibilidad es 0. De acuerdo con esto, *g-mean* es muy estricta cuando una clase se clasifica de forma completamente incorrecta.

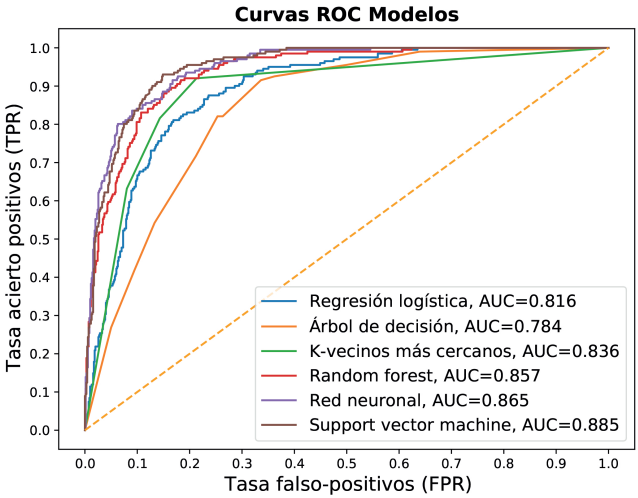
Una de nuestras medidas favoritas es el área bajo la curva o AUC *(Area Under the Curve)*. Esta medida se obtiene en base a la curva ROC *(Receiver Operating Characteristic)*, la cual grafica el desempeño (balanceado) del modelo para todos los posibles umbrales.

Recordemos la tercera etapa de la regresión logística, en donde debíamos elegir el umbral para nuestra salida probabilística. En principio, existen infinitos umbrales que podemos considerar entre 0 y 1. Cada umbral nos entrega un desempeño predictivo (sensibilidad y especificidad). En riesgo de crédito, podemos partir con el umbral 0, es decir, aceptar a todos (0% sensibilidad y 100% especificidad). Luego, vamos subiendo de a poco el umbral, rechazando algunos clientes, hasta llegar al umbral 1, es decir, rechazar a todos (100% sensibilidad y 0% especificidad).

La curva ROC gráfica en el eje Y la sensibilidad (acierto en la clase positiva) para todos los posibles umbrales, mientras que en el eje X tenemos la tasa de falsos positivos (conocida también como la tasa de «falsas alarmas»). Un buen modelo predictivo trazará una curva que estará cerca de la esquina superior izquierda, donde llegamos a un 100% de acierto en la clase positiva y 0% de error en el acierto de clase negativa. Notar que todas las posibles curvas parten en la esquina inferior izquierda y terminan en la esquina superior derecha, independiente de su desempeño.

Si queremos comparar el desempeño entre una regresión logística y un árbol de decisión, podemos ilustrar ambas curvas de desempeño en un gráfico único, para de esta forma tener una noción del desempeño predictivo. Esto se puede ver en la figura 6.9 para un ejemplo de riesgo de crédito (el presentado en los laboratorios de Python dentro de este libro).

Figura 6.9: Ejemplo curva ROC




En la figura 6.9, es posible ver las curvas ROC de una serie de modelos predictivos, incluyendo además el desempeño de un modelo al azar (línea punteada). El modelo más cercano a la esquina superior izquierda es *Support vector machine*, siendo entonces el de mejor desempeño.

Sin embargo, nuestro deseo es tener una medida única para comparar modelos más que un gráfico, que se puede complejizar si comparamos muchos modelos y/o las diferencias entre ellos son menores (como es el caso en la figura 6.9).

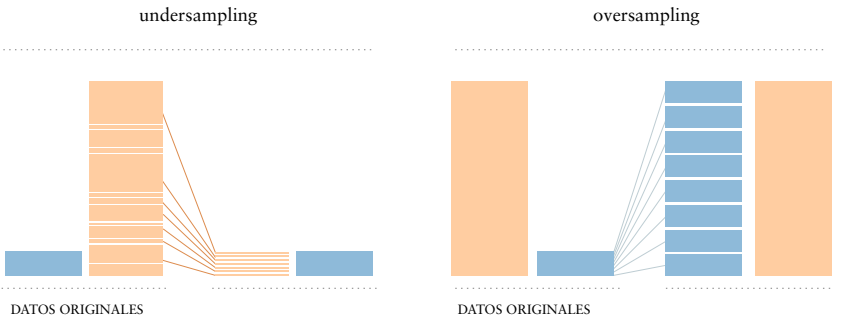
La medida AUC se obtiene, como su nombre lo dice, calculando el área bajo cada una de las curvas ROC. El valor máximo del AUC es 1, ya que es un cuadrado de arista 1. Si bien el mínimo teórico es 0, esto supondría que es un modelo predictivo perfecto si simplemente invertimos nuestro pronóstico. Esto es irrealista. Un valor de AUC de 0.5 corresponde a un modelo que decide al azar, es decir, que el 50% de las veces acierta (de forma balanceada) y el 50% restante falla. El AUC para cada modelo de la figura 6.9 se observa en la leyenda del gráfico.

## 6.8 Desbalance de clases

Es común que en las distintas aplicaciones de negocios donde se utilizan herramientas predictivas, nos encontremos con bases de datos desbalanceadas. Esto puede generar un problema no solo en la evaluación de modelos, sino también en su propia construcción. Para corregir este problema, existen soluciones mediante preprocesamiento de datos, soluciones en el entrenamiento de modelos y soluciones asociadas a la predicción (Maldonado *et al.*, 2019).

Sobre las soluciones de preprocesamiento, la idea es manipular el conjunto de datos de entrenamiento (y solo los de entrenamiento, nunca los de testeo), de tal forma que el modelo vea un conjunto de datos relativamente balanceado. Esto se puede hacer sacando observaciones de la clase mayoritaria *(undersampling)*, o bien generando objetos de forma sintética para la clase minoritaria *(oversampling)*. Ambos enfoques se presentan en la figura 6.10.

Figura 6.10: Diferencia entre *undersampling* y *oversampling*




El enfoque más común para realizar *undersampling* es hacerlo de forma aleatoria, es decir, seleccionar un porcentaje de datos de la clase mayoritaria al azar. Para *oversampling,* en cambio, el enfoque más común se llama SMOTE *(Synthetic Minority Over-sampling Technique)*. A grandes rasgos, esta técnica genera nuevos datos de la clase minoritaria interpolando observaciones existentes de esta clase (Maldonado *et al.*, 2019).

El segundo enfoque es construir modelos que favorezcan una clase por sobre la otra dentro del conjunto de entrenamiento (sin manipularlo). Esto se conoce como **aprendizaje sensible al costo** *(cost-sensitive learning)*. Esta área es más bien académica, por lo que no se discutirá en este libro.

Finalmente, el tercer enfoque es corregir la predicción del modelo cuando esta favorece a la clase mayoritaria. Esto se hace después que el modelo haya sido construido en base a la salida probabilística.

En palabras simples, los modelos establecen un umbral de 0.5 para clasificar una observación según «clase 1» y «clase 2». Por ejemplo, cuando decimos que un cliente es predicho como «clase 1», lo que estamos diciendo es que existe una probabilidad sobre el 0.5 de que ese cliente sea «clase 1». Independiente de que el cliente posea 99% de probabilidad o 51% de probabilidad, este de todas formas será predicho como «clase 1». No obstante, un cliente con 49% de probabilidad de ser «clase 1», el modelo lo predecirá como «clase 2», de la misma forma lo hará con un cliente con probabilidad 1%. Por lo tanto, podemos apreciar que manipulando este umbral 0.5 podemos hacer que el modelo sea más exigente para predecir cierta clase y más generoso para predecir la otra. Este enfoque se conoce como *thresholding*.

Supongamos para el caso aplicado de fuga de clientes, que nuestro modelo escogido con umbral 0.5 predice un 5% fugados y un 95% de no fugados. Sin embargo, nuestra base de datos histórica nos señala que un 10% de los clientes dejan la compañía, mientras que el 90% se mantiene. En este caso sería apropiado manipular el umbral de tal forma que el modelo sea forzado a predecir, por ejemplo, un 10% de fuga y un 90% de no fuga. Si bien respetar, al menos, la proporción original es recomendable, podemos ir incluso más allá asumiendo que el beneficio de predecir correctamente los fugados es mayor que con los no fugados. Por ejemplo, podemos tener presupuesto para una campaña de marketing que nos permita contactar al 15% de nuestros clientes, y en ese caso movemos el umbral de tal forma que el 15% de ellos sean predichos como fugados. Volveremos a este punto en el capítulo de *prescriptive analytics*.

## Referencias

Baesens, B. (2014). *Analytics in a Big Data World*. New Jersey: John Wiley and Sons.

Baesens, B., Roesch, D. & Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. New Jersey: John Wiley and Sons.

Han, J., Kamber, M. & Pei, J. (2011). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Maldonado, S., López, J. & Vairetti, C. (2019). «An alternative SMOTE oversampling strategy for high-dimensional datasets». *Applied Soft Computing*, 76, 380-389.

Thomas, L. C. (2009). *Consumer Credit Models: Pricing, Profit and Portfolios*. Oxford - Oxford Scholarship Online.

Verbeke, W., Baesens, B. & Bravo, C. (2017). *Profit Driven Business Analytics*. New Jersey: Wiley.

# Aplicación 3: Fuga de clientes

La fuga de clientes es una de las tareas predictivas más populares de la analítica de negocios, enfocada en identificar aquellos clientes que con mayor probabilidad dejarán una compañía **voluntariamente**. Una vez que estos clientes con potencial de fuga son detectados, la compañía realiza una campaña de retención focalizada en ellos para motivarlos a permanecer (Baesens, 2014).

La lealtad de los clientes es extremadamente beneficiosa: se estima que un incremento de un 5% en la tasa de retención puede lograr hasta un 18% de reducción en los costos operacionales. Focalizar recursos en los clientes candidatos a fuga en vez de abordar campañas masivas reduce el gasto de marketing (Fleming & Asplund, 2007).

Se estima que los clientes leales y comprometidos con la empresa generan 1.7 veces más utilidades que otros clientes (Fleming & Asplund, 2007). Dado esto, el esfuerzo en retención suele traer más réditos que los recursos invertidos en adquisición de nuevos clientes. De todos modos, es importante considerar esfuerzos en ambos frentes mediante una adecuada administración del presupuesto de marketing (Blattberg *et al.*, 2008).

Antes de detallar las principales técnicas para la detección de clientes con potencial de fuga, es importante posicionar esta tarea dentro de las diferentes actividades de marketing cuantitativo, donde los modelos predictivos pueden ser útiles.

## 1. CRM analítico

La administración de relaciones con el cliente (*customer relationship management* o CRM) es un enfoque que busca gestionar la interacción de la empresa y sus clientes vigentes y potenciales con

el fin de lograr relaciones más rentables y duraderas (Blattberg *et al.*, 2008).

Es importante hacer la desambiguación del término CRM con el *software* de igual nombre. Este tipo de herramientas computacionales corresponden a sistemas informáticos de apoyo a las decisiones de negocio. El *software* de CRM incluye varias funcionalidades, siendo la gestión de las relaciones con los clientes una de ellas.

Dentro de las tareas del enfoque CRM se incluye el análisis de diferentes fuentes de datos de los clientes con el fin de poder tomar mejores decisiones de forma informada. La utilización de modelos de *business analytics* se suele enmarcar en este concepto. Alternativamente, otro concepto similar es *database marketing* (Blattberg *et al.*, 2008).

Las tareas principales del CRM analítico son la adquisición de nuevos clientes, ventas cruzadas *(cross-selling)* y la retención de clientes. Con respecto al primer tema, el objetivo es identificar aquellos clientes que tienen una mayor probabilidad de entrar a la compañía y/o aquellos más valiosos para ella. Por ejemplo, modelos predictivos podrían ser capaces de predecir la probabilidad de que un cliente acepte una oferta focalizada en base a los datos históricos de los clientes que son o han sido parte de la compañía (Blattberg *et al.*, 2008).

La adquisición de nuevos clientes está directamente ligada con el concepto del valor del cliente (LTV), una fórmula que busca cuantificar a los clientes y llevar al presente los flujos futuros que ellos pueden generar a la empresa. Este concepto se formalizará más adelante dentro de este capítulo. Adicionalmente, el uso de modelos de *customer profiling* (ver Aplicación 1) pueden ser muy útiles para identificar segmentos de clientes que pueden ser más o menos atractivos para la compañía. En general, el cálculo del LTV se realiza a nivel individual o segmento, ya que los clientes suelen ser heterogéneos en términos de su valor en la gran mayoría de las empresas y sectores industriales (Maldonado & Domínguez 2021).

El LTV es un *input* importante para la implementación de la llamada **curva de adquisición**, la cual relaciona el valor del cliente

con los costos de adquisición (Blattberg *et al.*, 2008). Este tipo de estrategias permite definir el presupuesto adecuado para desarrollar programas de adquisición de clientes, para los cuales los modelos predictivos pueden ser una pieza fundamental.

La segunda tarea del CRM analítico corresponde a aumentar el consumo de los clientes vigentes y, por consiguiente, su valor. Por un lado, las ventas cruzadas *(cross-selling)* implican ofrecer productos o servicios que ofrece la empresa y que el cliente no posee. Por otro lado, ventas adicionales *(up-selling)* corresponde a «venderles más» de el o los productos o servicios que ya posee. Esto último puede realizarse mediante *upgrades* de servicio; por ejemplo, planes de telefonía móvil que incluyan un mayor número de minutos y/o de consumo de datos *(gigabytes)*.

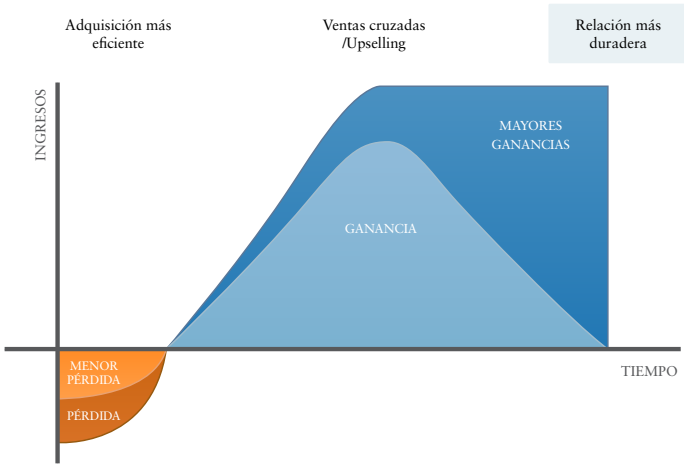
Existen diferentes estrategias predictivas que pueden ser útiles para mejorar las ventas cruzadas. Un enfoque es el uso de modelos de clasificación para predecir la próxima compra del cliente (*next product to buy* o NPTB). Esto se puede hacer con modelos tradicionales o de caja negra (Blattberg *et al.*, 2008). Otra pregunta de negocios común es **cuándo** el cliente hará su próxima compra, lo que es una tarea de regresión que suele abordarse con técnicas de análisis de supervivencia (ver capítulo 9).

En el capítulo 3 se discutieron técnicas de reglas de asociación y filtrado colaborativo que también suelen ser consideradas para ventas cruzadas. Empresas como Amazon usaron este tipo de técnicas para ofrecer sugerencias de productos tomando como base los considerados por el o la cliente.

La figura App3.1 resume las tres tareas principales del CRM analítico en base al ciclo de vida del cliente. Este modelo conceptual ilustra el valor de un cliente promedio a lo largo del tiempo. A la izquierda de la figura App3.1 se indica que los clientes primero generan costos para una empresa por concepto de adquisición, por lo tanto, su valor es negativo. A medida que el cliente crece y madura en la empresa, su valor aumenta. Finalmente, el cliente pasa por una etapa de pérdida de compromiso *(disengagement)* hasta que termina dejando la compañía.

En el medio de la figura App3.1 se muestra el potencial impacto de *business analytics* en las decisiones de CRM. Primero, una adquisición efectiva de clientes reduce las pérdidas iniciales, focalizando los recursos en aquellos clientes más valiosos y/o con mayor probabilidad de entrar a la compañía. Segundo, estrategias de *cross-* y *up-selling* permiten aumentar el valor del cliente durante su etapa de maduración. Finalmente, estrategias proactivas de lealtad que busquen retener a los clientes valiosos buscan alargar su etapa de madurez y evitar el *disengagement* (lado derecho de la figura App3.1).

Figura App3.1: Principales del CRM analítico en base al ciclo de vida del cliente




## 2. LTV y beneficio de una campaña de retención

El marketing necesita métricas si desea ser relevante para la alta gerencia y no quedar solo en un relato de buenas intenciones. Más allá de las ya tradicionales ventas y participación de mercado, el LTV se posiciona como una de las medidas más importantes (Blattberg *et al.*, 2008).

El LTV, por un lado, es una medida que funciona por sí sola, y permite hacer un diagnóstico de la «salud» del negocio. El LTV (promedio o por segmento) puede formar parte de una batería de métricas que permitan generar un diagnóstico a largo plazo. Por

otro lado, es un *input* para otros modelos táctico-operativos que permitan tomar decisiones a nivel de cliente o segmento (Maldonado & Domínguez, 2021).

A nivel individual, corresponde al valor presente neto de las utilidades (beneficios menos costos) generadas por un cliente *j*. Se calcula de la siguiente manera (Blattberg *et al.*, 2008):

Ecuación App3.1

$$LTV_j = \sum_{t=1}^{T} \frac{(Beneficios_{j,t} - Costos_{j,t}) \cdot r_t}{(1+\delta)^t}$$

Donde $\delta$ es la tasa de descuento, que se utiliza para penalizar los flujos futuros por sobre los más recientes, y $T$ es el horizonte de tiempo. El parámetro $r_t$ corresponde a la tasa de retención, es decir, la probabilidad de que un cliente se quede en la compañía en un cierto instante (Maldonado & Domínguez, 2021).

Con respecto a la tasa de descuento, se suele considerar el *weighted average cost of capital* (WACC) como costo de oportunidad del dinero. Si bien la fórmula del WACC considera información relativa a la empresa, como fondos propios, deudas y retornos, existen tabulaciones con valores promedio del WACC a nivel país y sector industrial que se pueden considerar. Un ejemplo es el cálculo realizado por la Universidad de Nueva York (NYU) para los WACC promedio para Estados Unidos[13].

El horizonte $T$ puede ser un periodo fijo o una perpetuidad ($T = \infty$). Por ejemplo, se ha propuesto $T = 3$ años para la industria bancaria como un valor esperado de vida del cliente, además de $\delta$ igual a la tasa de inflación (Maldonado & Domínguez, 2021).

Volviendo a la fuga de clientes, esta tarea suele abordarse como un problema de clasificación binaria, realizando predicciones para el próximo periodo con el fin de anticipar si un cliente va a dejar, o no, la compañía. Para esto se considera un mes, un trimestre o un año de estudio, dependiendo de las decisiones que se busque tomar

[13] http://people.stern.nyu.edu/adamodar/New_Home_Page/datafile/wacc.htm

(Maldonado, López y Vairetti, 2020). Alternativamente, se puede responder la pregunta ¿cuándo el cliente dejará la compañía? Este problema se suele abordar con análisis de supervivencia, al igual que la tarea de incidencia de compra.

A diferencia de la aplicación de riesgo de créditos, fuga de clientes es una aplicación donde no hay aspectos regulatorios que considerar en el sentido de los modelos predictivos a utilizar. Por esto, hay libertad para elegir el mejor modelo predictivo de forma empírica mediante validación de modelos (Verbeke *et al.*, 2017). Esto incluye la utilización de modelos de tipo caja negra. Por ende, es una excelente aplicación para motivar nuestro capítulo de teoría siguiente, nuestro capítulo de teoría siguiente.

Al igual que en *credit scoring*, la predicción de fuga sufre del problema del desbalance de clases ya que es común encontrar conjuntos de datos con menos de un 5% de clientes fugados, sobre todo cuando se considera una ventana de un mes. Se proponen las mismas soluciones que en la aplicación de riesgo de crédito, aunque es más común considerar SMOTE como técnica avanzada para aumentar artificialmente la cantidad de ejemplos de la clase fuga por sobre la utilización de ponderaciones por clase (Verbeke *et al.*, 2017).

A la libertad de elección de modelos de clasificación y el problema de desbalance de clases, es necesario sumar el hecho de que los costos de clasificación son asimétricos: resulta mucho más valioso lograr retener un cliente con potencial de fuga que «desperdiciar» la campaña de retención en un cliente leal[14]. Debido a estos tres factores, es clave contar con una estrategia de validación orientada al objetivo, que es el éxito de la campaña de retención. Aquí es donde entra en acción el LTV (Verbeke *et al.*, 2017).

La dinámica de una campaña de retención suele ser la siguiente: una fracción $\eta$ de los $N$ clientes actuales es contactada (aquellos que se estiman con mayor propensión de fuga). El costo de contactar al cliente se asume de $f$, mientras que el costo del incentivo ofrecido

[14] Es cierto que la entrega de un beneficio a un cliente suele tener una buena acogida, por más que este no tenga intenciones de fugarse. De todos modos, los modelos asumen la simplificación de que este gesto no trae beneficios asociados, solo costos.

se asume de $d$. Dentro de los clientes contactados, algunos quieren realmente abandonar la compañía y otros no. Para este último grupo se asume que todos tomarán el incentivo y se quedarán, ya que no tenían ninguna intención de dejar la compañía. Para el primer grupo, sin embargo, se asume que pueden pasar dos cosas:

- Existen clientes que aceptarán el incentivo y esto los convencerá a quedarse en la compañía. De esta forma se gana el *LTV* de los clientes, que se asume mucho más alto que los costos variables de campaña. Se asume que una fracción $\gamma$ de los clientes que quieren abandonar logra ser retenida.
- La fracción restante $(1 - \gamma)$ no logra ser convencida, a pesar del esfuerzo de retención. Se asume como pérdida *f*, el costo de contacto, pero no *d* ya que el incentivo es rechazado.

De acuerdo con Neslin *et al.* (2017), la utilidad de una campaña de retención se cuantifica de la siguiente manera:

Ecuación App3.2

$$Profit = N\eta[(\gamma LTV + d(1-\gamma))\pi_1\lambda - d - f] - A$$

Donde $\lambda$ es el coeficiente de *lift*, que se interpreta como qué tanto más probable es que el grupo contactado de clientes se fugue en relación con el total de los clientes. Este coeficiente se obtiene en base al modelo predictivo. Asumiendo que $y = 1$ es un cliente fugado, $\pi_1$ representa la fracción del total de clientes que se fugan. Finalmente, $A$ es el costo fijo de la campaña (Verbeke *et al.*, 2017).

La fórmula de Neslin *et al.* ha sido adoptada no solo para la evaluación de modelos predictivos, sino también para guiar el aprendizaje de los modelos de clasificación (Verbeke *et al.*, 2017; Maldonado, López y Vairetti, 2020). Esta vinculación entre modelación predictiva y toma de decisiones es un buen ejemplo de *prescriptive analytics* y será discutida con más detalle en el capítulo 10.

Para finalizar, es importante notar que la tarea de predicción de fuga debe lidiar con ambientes cambiantes en sectores como telecomunicaciones o banca. Esto ocurre por diversas razones. Primero, la

tecnología evoluciona y la competencia entre los diferentes actores suele variar en base a tecnologías emergentes. Segundo, las campañas de retención varían en el tiempo en base a la tecnología y la competencia. Finalmente, el impacto de una campaña de retención suele variar en el tiempo (en general, su efecto es decreciente). Estos cambios suelen causar obsolescencia en los modelos predictivos, que deben ser constantemente monitoreados para evaluar que la capacidad predictiva se mantenga en el tiempo. En el capítulo 11 se discutirá con mayor detalle el monitoreo de modelos una vez que fueron implementados.

## Referencias

Baesens, B. (2014). *Analytics in a Big Data World*. New Jersey: John Wiley and Sons.

Blattberg, R. C., Kim, B. D. & Neslin, S. A. (2008). *Database Marketing: Analyzing and Managing Customers*. New York: Springer.

Fleming, J. & Asplund, J. (2007). *Human Sigma: Managing the Employee-Customer Encounter.* New York: Gallup Press.

Maldonado, S., Domínguez, G., Olaya, D. & Verbeke, W. (2021). «Profit-driven Churn Prediction for the Mutual Fund Industry: a Multisegment Approach». *Omega,* 100, ID 102380.

Maldonado, S., López, J. & Vairetti, C. (2020). «Profit-based Churn Prediction based on Minimax Probability Machines». *European Journal of Operational Research,* 284(1), 273-284.

Verbeke, W., Baesens, B. & Bravo, C. (2017). *Profit Driven Business Analytics*. New Jersey: Wiley.

# 7. *Predictive Analytics* II: Modelos «caja negra» tradicionales

La analítica de negocios predictiva se basa en buena parte en la utilización de modelos transparentes tradicionales como los descritos en el capítulo anterior. Sin embargo, existen tareas bastante complejas, como *fraud analytics,* en donde estamos dispuestos a sacrificar esta interpretabilidad en pos de tener una mejora en el desempeño predictivo (Baesens *et al.*, 2015).

En este capítulo se introducen los modelos de aprendizaje supervisado *K* vecinos más cercanos, *Naive Bayes* y *Random Forest*. Estos tres modelos permiten complementar nuestra caja de herramientas de modelos predictivos con el fin de poder tomar la mejor decisión, en términos de modelación, para cualquier tarea predictiva. Nuestra caja de herramientas quedará completa con el capítulo siguiente, donde exploraremos redes neuronales tradicionales, máquinas de soporte vectorial y aprendizaje profundo.

Si bien con estos modelos no es posible dilucidar el proceso que siguió el algoritmo para llegar a la toma de decisiones, existen diferentes estrategias que permiten «abrir» la caja negra e identificar, al menos, cuáles fueron las variables que más contribuyeron en este proceso. Dentro de las estrategias interpretativas más populares se encuentran BORUTA y TreeSHAP, las cuales serán formalizadas en este capítulo. Se mencionarán los conceptos más importantes, acompañados con una guía aplicada para la construcción y validación de cada uno de los modelos.

Este capítulo finaliza con una discusión de cómo extender estos modelos a una clasificación multiclase. Si bien la mayoría de los problemas de clasificación en *business analytics* son de tipo binario (cliente buen o mal pagador, fuga o no fuga, transacción fraudulenta

o no, etcétera), ciertas tareas pueden considerar diferentes categorías como variable objetivo (Verbeke *et al.*, 2017). Algunos ejemplos son la predicción de la clasificación de riesgo de una empresa (AAA, AA, A, BBB, BB, B, C, D y E) (Baesens *et al.*, 2016) o la predicción del nivel educacional de una persona.

## 7.1 *K* vecinos más cercanos

En 1951, Evelyn Fix y Joseph Lawson Hodges Jr., introdujeron por primera vez el principio básico detrás de este modelo: el comportamiento de un objeto nuevo puede ser predicho en base a sus vecinos más cercanos.

El modelo conocido como K vecinos más cercanos (K *nearest neighbors*, o k-NN) reconoce similitudes entre las observaciones de la base de entrenamiento y el valor desconocido, para luego identificar los *K* vecinos más cercanos. Cada observación representa un punto en el espacio de n-dimensiones (los atributos), lo que permite ubicar dentro de un espacio al valor desconocido para agruparlos en un *cluster* compuesto por K ejemplos del conjunto de entrenamiento.

Para determinar la distancia entre el valor nuevo y los K vecinos, en general se utiliza la distancia Euclidiana, donde se suma las diferencias al cuadrado en cada una de las dimensiones presentes (la misma que utilizamos en *clustering* en el capítulo 3). Dadas dos observaciones $X = (x_1, x_2, \ldots, x_n)$ *e* $Y = (y_1, y_2, \ldots, y_n)$, esta distancia se define como:

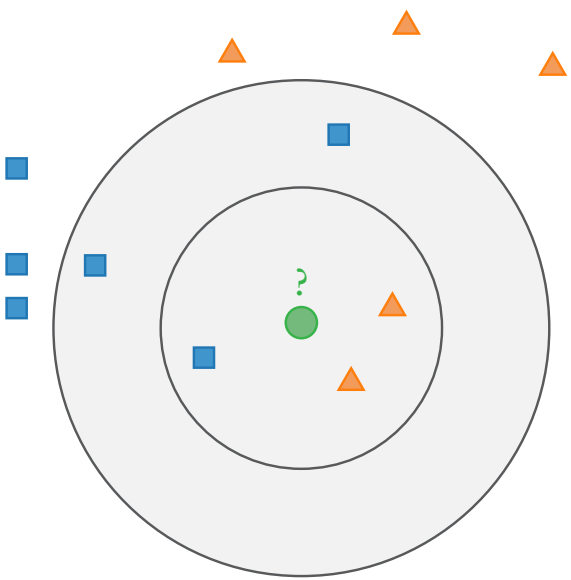
Ecuación 7.1

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Por consiguiente, el valor desconocido es asignado dependiendo de la clase de los K vecinos más cercanos, obteniendo el valor de la clase que más se repita (la moda). Alternativamente, se puede utilizar la distancia de Manhattan formalizada en el capítulo 3.

La cantidad de vecinos más cercanos alrededor del valor desconocido K se define al momento que se construye el modelo y representa su único parámetro. Por ejemplo, en la figura 7.1 podemos observar dos clases (cuadrados y triángulos) y un ejemplo nuevo de clase desconocida (el círculo). En el caso de elegir K=3, entonces se podría decir que el nuevo ejemplo corresponde a un triángulo con una probabilidad de 66% o $\frac{2}{3}$. Por otro lado, elegir K=5 implica «ampliar el círculo» para seleccionar más vecinos. En este caso, la decisión cambia y el método sugiere que este nuevo punto sea un cuadrado. Este ejemplo ilustra la dependencia del método con relación al parámetro K. Esto no significa que una solución sea mejor que la otra necesariamente, ya que para ello precisamos evaluar cada una bajo un esquema de validación adecuado (ver capítulo anterior).

Figura 7.1: Funcionamiento K vecinos más cercanos



Se sugiere considerar un parámetro K impar para evitar empates. En la figura 7.1, por ejemplo, si hubiésemos elegido K=4 se generaría un empate entre los K vecinos si es que el 50% de ellos son triángulos y el 50% cuadrados.

No existe un número de vecinos óptimo que sea capaz de adaptarse a todas las situaciones. Dicho de otro modo, cada contexto es distinto al otro, en términos de datos principalmente, por ende, no existe un número óptimo de K vecinos predefinido por un marco teórico, sino que dependerá del desempeño obtenido con los datos usados. Cabe destacar que lo más común es considerar K=5 por defecto, pero más por una razón de comodidad para evitar explorar muchos valores diferentes que por un argumento teórico sólido (James *et al.*, 2017).

La utilización de la distancia Euclidiana implica que cada atributo recibe exactamente la misma ponderación en el cálculo del vecindario (notar que la sumatoria en la ecuación 1 no entrega pesos a cada dimensión). Es un supuesto bastante «ingenuo» asumir que todas las variables son igualmente importantes. La regresión logística, por ejemplo, era capaz de encontrar las ponderaciones óptimas para cada variable. Esto trae como consecuencia un posible mal desempeño de k-NN, sobre todo en caso de existir muchas variables irrelevantes con relación a las relevantes.

Es muy recomendable realizar algún tipo de filtro de selección de atributos antes de utilizar k-NN (ver capítulo 2), dado que el modelo no tiene mecanismo alguno de priorización de variables. Además, es muy importante normalizar los datos y llevarlos a una escala común para realizar un cálculo de distancias adecuado.

En resumen, los aspectos positivos del modelo son su simplicidad, su fácil implementación y el hecho de que solo tenga un parámetro calibrable. Sin embargo, cuenta también con algunos aspectos negativos, como la incapacidad de interpretar el modelo y su supuesto de similar importancia entre variables.

Aspectos importantes a considerar:

- Utiliza la definición de un vecindario como método de clasificación.
- No es un modelo interpretable.
- Se debe considerar un valor K impar para evitar empates. Este es el único parámetro calibrable, aunque algunas implementaciones permiten variar la medida de distancia (Euclideana o Manhattan).
- El modelo considera que todas las variables son relevantes e igual de importantes, lo que puede afectar su desempeño ya que, en general, no se cumple en la realidad.
- Se debe normalizar/escalar la base de datos para ejecutar el modelo correctamente.
- Modelo diseñado para variables numéricas.

## 7.2 Naive Bayes

Naive Bayes es un clasificador estadístico que predice la probabilidad de pertenencia de una clase. Asume que todas las variables son independientes entre sí, lo que rara vez se cumple en la práctica. Debido a este supuesto «ingenuo» es que el modelo recibe su nombre[15].

Pese a ello, hay estudios que señalan que es comparable el desempeño de este modelo con el árbol de decisión o redes neuronales. Además, este modelo ha demostrado un alto nivel de precisión y velocidad aplicado en Big Data. Es uno de los algoritmos más simples y poderosos para la clasificación basado en el teorema de Bayes con una suposición de independencia entre los predictores, para un conjunto de datos muy grandes (Han *et al.*, 2011).

[15] *Naïve* o *naive* significa, en español, «ingenuo» o «inocente».

Naive Bayes está basado en el teorema de Bayes, propuesto por el matemático británico Thomas Bayes, del siglo XVIII. Este teorema determina la probabilidad condicional, definida como la probabilidad de que ocurra un resultado, basado en un resultado previo. Asimismo, proporciona una forma de revisar las predicciones dada las nuevas evidencias.

*Teorema de Bayes:*

Sea X una muestra de datos donde la clase es incierta, y h como algunas hipótesis, tal como que la muestra de datos X pertenece a una clase C específica.

- **P(h|X)**: es la probabilidad de la hipótesis h dada los datos de X (probabilidad posterior). Por ejemplo, supongamos que el conjunto de datos X consta de frutas y verduras, describiendo el color y la forma. Supongamos que la observación X es roja y redonda, y que la hipótesis de H es que X es un tomate. Entonces P(h|X) se define como la probabilidad de que la observación X sea un tomate dado que X es rojo y redondo.
- **P(h)**: es la probabilidad de que la hipótesis h sea cierta (independientemente de los datos) (probabilidad previa de h). Por ejemplo, sería la probabilidad de que cualquier muestra de datos sea un tomate independientemente de cómo se vea la data. Es independiente de X.
- **P(X|h)**: es la probabilidad de los datos X dado que la hipótesis h era cierta. (probabilidad posterior). Como ejemplo, sería la probabilidad de que X es roja y redonda, dado que sabemos que D es efectivamente un tomate.
- **P(X)**: es la probabilidad de los datos (independientemente de la hipótesis) (probabilidad previa). Como ejemplo, sería la probabilidad que la muestra de datos de nuestro set de frutas y verduras sea roja y redonda.

Las probabilidades se estiman de la siguiente manera: por un lado, P(X), P(h) y P(X|h) deben ser estimados dado la muestra de datos recibida. Para luego, por medio del teorema de Bayes, obtener la probabilidad posterior *P(h|X)* dada por:

Ecuación 7.2

$$P(H|X) = \frac{P(X|H)\,P(H)}{P(X)}$$

*Clasificador Bayesiano Naive:*

¿Cuáles son los pasos del clasificador? A continuación, se indican los cuatro pasos a seguir:

1.- Supongamos que tenemos dos tipos de clases —C1 y C2— y un nuevo ejemplo X a predecir (sin etiqueta). El modelo va a predecir la clase de X según la mayor probabilidad *a posteriori* que tiene la observación, dado los atributos presentes en X, es decir:

Ecuación 7.3

$$P(C_i|X) > P(C_j|X)\ para\ j = [1;2], j \neq i$$

Ahora, invertiremos el problema utilizando el teorema de Bayes. En otras palabras, buscaremos cuál de las dos clases es más compatible con los datos en X:

Ecuación 7.4

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

2.- Como sabemos, $P(X)$ es constante para todas las clases, dado que es la probabilidad de que se de la combinación de variables en X. Así, solo necesitamos evaluar $P(X|C_i)P(C_i)$, para lograr obtener la probabilidad posterior $P(C_i|X)$.

Los cálculos de $P(C_i)$ son bastante simples ya que son las probabilidades *a priori* de cada clase i, es decir, la fracción de clientes

en el conjunto de entrenamiento que pertenecen a cada clase (sin saber nada de sus variables).

3.- Para calcular $P(X|C_i)$ haremos el supuesto Naive: todas las variables son independientes entre sí. De esta forma, la probabilidad de que un cliente tenga un vector de características X equivale a la multiplicación de las probabilidades asociadas a cada variable.

4.- Finalmente, la clase $C_i$ es asignada a la observación X, si y solo si:

Ecuación 7.5

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j)\ para\ j = [1;2], j \neq i$$

Este modelo es más simple de comprender con un ejemplo numérico. Volvamos a la aplicación para predecir si un cliente comprará una *tablet*, o no, discutida en el capítulo anterior. Los datos se presentan nuevamente en la figura 7.2:

Figura 7.2: Datos de entrenamiento para compra de *tablet*

| ID | Edad | Nivel de ingreso | Universitario | Deuda pendiente | Clase: compra *tablet* |
|---|---|---|---|---|---|
| 1 | <25 | medio | no | sí | no |
| 2 | <25 | bajo | sí | no | sí |
| 3 | >40 | medio | sí | sí | sí |
| 4 | <25 | medio | sí | no | sí |
| 5 | 25-40 | medio | no | no | sí |
| 6 | 25-40 | alto | sí | sí | sí |
| 7 | >40 | medio | no | no | no |
| 8 | <25 | alto | no | sí | no |
| 9 | <25 | alto | no | no | no |
| 10 | 25-40 | alto | no | sí | sí |
| 11 | >40 | medio | no | no | no |
| 12 | >40 | bajo | sí | sí | sí |
| 13 | >40 | bajo | sí | no | no |
| 14 | 25-40 | bajo | sí | no | sí |
| 15 | <25 | medio | sí | sí | sí |

La tarea predictiva en estudio considera los siguientes atributos: Edad, Ingreso, Universitario y Deuda_Pendiente. La variable objetivo está definida como «Compra_Tablet», compuesta de dos valores (sí; no). Donde $C_1$ corresponde a la clase Compra_Tablet = sí, y $C_2$ corresponde a Compra_Tablet = no. Para este caso, la muestra desconocida que queremos clasificar es la siguiente:

Ecuación 7.6

$$X = (Edad = " > 40", Ingreso = Medio, Universitario = no, Deuda_Pendiente = si$$

Para esto, necesitamos maximizar $P(X|C_i)P(C_i)$, donde $i = 1, 2$. (paso 1).

El paso 2 consiste en calcular los *priors* $P(C_i)$, es decir, la probabilidad previa de cada clase. Esta se puede calcular en base a la muestra de entrenamiento (9 de 15 clientes compran y los 6 restantes no lo hacen):

Ecuación 7.7

$$P(Compra_Tablet = si) = \frac{9}{15} = 0{,}6$$

$$P(Compra_Tablet = no) = \frac{6}{15} = 0{,}4$$

El paso 3 consiste en obtener $P(X|C_i)$, donde $i = 1, 2$. De acuerdo con el supuesto Naive, estudiaremos los valores de X de cada variable por separado y para ambas clases, para luego realizar las multiplicaciones correspondientes:

Ecuación 7.8

$$P(Edad = " > 40"|Compra_Tablet = sí) = \frac{2}{9} = 0{,}222$$

$$P(Edad = " > 40"|Compra_Tablet = no) = \frac{3}{6} = 0{,}5$$

$$P(Ingreso = Medio|Compra_Tablet = sı) = \frac{4}{9} = 0{,}444$$

$$P(Ingreso = Medio|Compra_Tablet = no) = \frac{3}{6} = 0{,}5$$

$$P(Universitario = no|Compra_Tablet = si) = \frac{2}{9} = 0{,}222$$

$$P(Universitario = no|Compra_Tablet = no) = \frac{5}{6} = 0{,}833$$

$$P(Deuda_Pendiente = si\,|Compra_Tablet = si) = \frac{5}{9} = 0{,}555$$

$$P(Deuda_Pendiente = si\,|Compra_Tablet = no) = \frac{2}{6} = 0{,}333$$

Ecuación 7.9

$$P(X|Compra_Tablet = si) = 0{,}222 \cdot 0{,}444 \cdot 0{,}222 \cdot 0{,}555 = 0{,}012$$

$$P(X|Compra_Tablet = no) = 0{,}5 \cdot 0{,}5 \cdot 0{,}833 \cdot 0{,}333 = 0{,}069$$

Dado los valores obtenidos, el paso 4 consiste en calcular $P(X|C_i)P(C_i)$ y elegir el mayor de ambos valores:

Ecuación 7.10

$$P(X|Compra_Tablet = si)P(Compra_Tablet = si) = 0{,}012 \times 0{,}6 = 0{,}007$$

$$P(X|Compra_Tablet = no)P(Compra_Tablet = no) = 0{,}69 \times 0{,}4 = 0{,}276$$

Según los resultados obtenidos, el modelo clasificador de Naive Bayes predice que «Compra_Tablet = no» para la base de muestra X (0,276>0,007). En otras palabras, esto indica que, dada la base de entrenamiento utilizado, cuando el individuo es mayor a 40 años, con un ingreso medio, no es universitario que posee una deuda pendiente, es más probable que el individuo no compre una *tablet*.

En resumen, los aspectos positivos de Naive Bayes son su simplicidad, su fácil implementación y el hecho de que no tenga parámetros calibrables. Algunos de sus aspectos negativos son la incapacidad de interpretar el modelo y su supuesto «ingenuo» de que todas las variables son independientes entre sí.

Aspectos importantes a considerar:

- Utiliza el teorema de Bayes como método de clasificación.
- No es un modelo interpretable.
- No tiene parámetros calibrables.
- El modelo considera que todas las variables son independientes, lo que puede afectar su desempeño ya que, en general, no se cumple en la realidad.
- Modelo diseñado para variables nominales. Se debe discretizar si se desea aplicar a variables numéricas.
- El modelo tiene algunos defectos que pueden hacer que falle. Por ejemplo, si la variable categórica tiene una categoría en el conjunto de datos de prueba que no se observó en el conjunto de datos de entrenamiento, el modelo asignará una probabilidad de 0 y no podrá hacer una predicción.

## 7.3 Random Forest

En 2001, Leo Breiman, estadístico estadounidense de la Universidad de California, Berkeley, publicó el primer artículo sobre *Random Forest* (RF). Este es un modelo multiclasificador o *ensemble*, dado que utiliza una combinación de distintos árboles de decisión, los que determinan la clasificación del valor incógnito a partir de la moda de las salidas de clase de los árboles. Se define como un método de ensamblado dado que usa múltiples modelos de analítica, específicamente árboles de decisiones, en vez de solo un modelo.

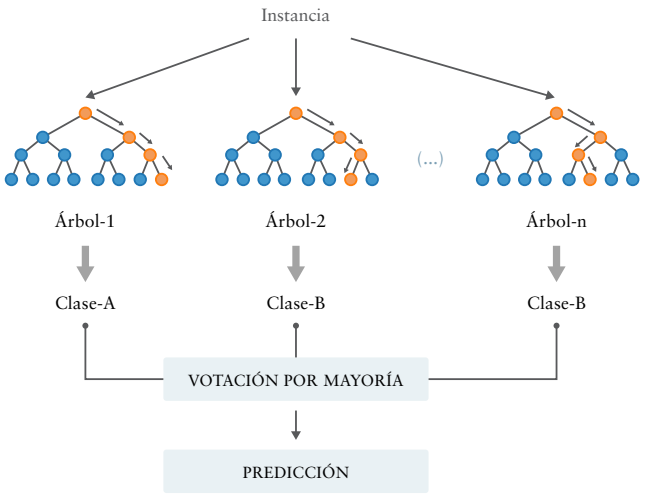
Multiclasificadores se define como un conjunto de modelos predictivos que se combinan por medio de votación simple u otro mecanismo más sofisticado. Esto permite tener una mejor precisión

en la predicción del modelo. Además, es más eficiente en su construcción dado que no utiliza necesariamente todos los atributos y/u observaciones, también porque permite ejecutar en paralelo bajo un conjunto extenso de datos[16].

El funcionamiento de RF nos recuerda a un Entmoot o reunión de *ents*. En el universo de J. R. R. Tolkien, los *ents* son criaturas con forma de árbol, pero con capacidades humanoides. En «Las dos torres», segundo volumen de *El señor de los anillos,* los *ents* deciden realizar un Entmoot en el bosque de Fangorn para discutir si van a la guerra, o no, contra Saruman (un mago antagonista). Un modelo de RF es esencialmente una serie de árboles de decisión que reciben una observación a predecir, entregan su pronóstico y, mediante votación, deciden la clase a la cual se asignará esta nueva instancia.

En la figura 7.3 se puede observar cómo se compone este modelo, en donde primero se construyen *n* árboles en base al conjunto de entrenamiento. Cada árbol recibirá filas y columnas diferentes de la matriz X, por lo que no serán iguales. Luego, las diferentes instancias del conjunto de prueba se procesan en cada árbol, que puede entregar etiquetas diferentes. La etiqueta final se define en base a votación por mayoría (la moda de las *n* etiquetas asignadas).

Figura 7.3: Esquema Random Forest




16 https://www.redalyc.org/pdf/925/92513106005.pdf

Parece evidente que el concepto bosque *(forest)* en el nombre RF se inspira en la construcción de muchos árboles de decisión. Veamos ahora el porqué del concepto «al azar» *(random)*.

Cada árbol se entrena con un subconjunto de observaciones y atributos elegidos al azar para así tener un set de árboles distintos. ¿Cuál es la finalidad de esto? Permite obtener distintas «opiniones» entre los árboles, a pesar de que sean árboles «débiles», en el sentido de que no son construidos con la mejor información.

¿Qué ocurre si cada árbol se constituye de todos los atributos y observaciones? No habría diversidad de «opiniones», generando redundancia entre los árboles, de tal forma que no se sacaría provecho del modelo. Mientras los árboles sean distintos, en promedio van a llegar a un buen resultado.

*Bagging* o *Bootstrap Aggregating* se denomina al método que asigna al azar observaciones para alimentar los árboles de decisiones. Este método corresponde a un simple muestreo con reemplazo de la matriz de datos, la construcción de modelos en los diferentes subconjuntos de datos y, finalmente, la votación por mayoría.

De forma similar, la selección de atributos es al azar. La cantidad de atributos que tendrá cada árbol para su construcción es un parámetro del problema, al igual que la cantidad de árboles a construir y el o los criterios de poda que se consideren para cada árbol.

En resumen, los aspectos positivos de *Random Forest* son su capacidad predictiva superior y su flexibilidad para capturar patrones complejos que le permite ser una excelente alternativa, tanto para datos numéricos, como nominales. Sus desventajas son la pérdida de interpretación al considerar la combinación de muchos árboles, la cantidad de parámetros a calibrar y el hecho de que, si se cuenta con muchos atributos irrelevantes, la mayoría de los árboles serán débiles y el modelo no tendrá el desempeño esperado.

Aspectos importantes a considerar:

- Combina diferentes árboles de decisión como método de clasificación.
- No es un modelo interpretable.
- Tiene, al menos, tres parámetros calibrables (número de árboles, número de atributos por árbol y criterios de poda), lo que complejiza encontrar su configuración óptima. De todos modos, el modelo es robusto y las configuraciones por defecto que consideran las diferentes implementaciones son lo suficientemente buenas como para no complicarnos demasiado con un proceso costoso de calibración.
- El proceso de *bagging* y selección al azar de atributos permite la construcción de los árboles en paralelo, acelerando el proceso de entrenamiento.
- Si bien se pierde la propiedad de hacer una selección de atributos empotrada que tenía el árbol de decisión (ver capítulo anterior), con RF es posible establecer la contribución de cada atributo en la solución final.
- Modelo diseñado para cualquier tipo de variable. Excelentes resultados predictivos.

Para finalizar el mundo de los multiclasificadores o *ensembles*, es importante destacar que existe una enorme variedad de alternativas que han surgido en las últimas decadas gracias al éxito de RF. Una de las más populares gracias a su velocidad de entrenamiento y desempeño predictivo es *eXtreme Gradient Boosting* (XGBoost), el cual suele tener un mayor acierto que RF ya que logra mejorar algunas de sus propiedades.

XGBoost es bastante similar a RF ya que combina varios arboles clasificadores que reciben conjuntos diferentes de variables y atributos. *Boosting* es muy similar a *bagging* ya que genera diferentes muestras con reemplazo, sin embargo, introduce datos que fueron mal clasificados por árboles previos en los árboles siguientes. Por ejemplo, si el Árbol 1 comete 100 errores de entrenamiento con los datos que recibe al azar, los datos que recibirá el Árbol 2 se seleccionarán al azar, pero estos 100 errores tendrán una mayor probabilidad de ser seleccionados que los datos que no estuvieron en el Árbol 1 o bien que fueron bien clasificados por él. *Adaboost* (ADAptative BOOSTing) es un modelo de *boosting* muy popular, ya que fue el primero que siguió esta idea.

Una de las principales desventajas de *Adaboost* en relación con RF es que el primero no puede paralelizarse ya que requiere del Árbol 1 y su desempeño para poder construir el Árbol 2. Aquí nace la idea de *gradient boosting* de construir una función de error a minimizar considerando variantes del Método del Gradiente. Esta estrategia de optimización permite un proceso de aprendizaje de errores muy eficiente, lo que le permite alcanzar excelentes resultados con rapidez. Adicionalmente, es posible distribuir este proceso en entornos Big Data como Hadoop o Spark. Esta estrategia de aprendizaje basada en gradiente será discutida en el siguiente capítulo, ya que es la base para la optimización de las redes neuronales artificiales. XGBoost es entonces la variante más popular de *gradient boosting* y una de las mejores alternativas de enfoques caja negra para llegar a buenos resultados sin gran esfuerzo.

## 7.4 Interpretando modelos de caja negra I: BORUTA

El algoritmo BORUTA corresponde a un método para generar un ranking de atributos considerando un modelo predictivo de caja negra de base. Este método fue diseñado originalmente en 2010 para ser utilizado con *Random Forest* y, por ende, se presenta a continuación de este. Pese a esto, BORUTA puede ser utilizado con otros clasificadores.

El ranking que genera BORUTA se construye evaluando el desempeño de diferentes subconjuntos de variables con un clasificador base. En este sentido, puede considerarse como un método de selección de atributos

envolvente o *wrapper* (ver capítulo 6). Una de las virtudes de BORUTA es que el algoritmo mismo nos entrega un umbral para la contribución individual de cada atributo. En otras palabras, BORUTA nos dice cuáles son los atributos más relevantes, los ordena de acuerdo con su contribución y nos sugiere eliminar aquellos que no aportan lo suficiente.

Una característica de BORUTA es que los atributos no compiten entre sí, sino con una versión aleatoria de ellos (con valores intercambiados entre filas). Veamos cómo funciona:

1. Primero, se duplica el conjunto de datos y se mezclan los valores en cada columna repetida. A estas versiones aleatorias de las columnas originales se les denomina **características sombra**.
2. Luego, se entrena un clasificador (normalmente RF) en el conjunto de datos. A través de este clasificador se obtiene la importancia de cada una de las variables (originales y «sombra»). En la medida que su puntuación sea mayor, la variable en cuestión resulta más importante.
3. El algoritmo evalúa la contribución de cada una de sus características originales y la compara con un umbral: la mayor contribución obtenida por una variable sombra. Los resultados son registrados como un vector de valores binarios, en donde se le denomina un acierto *(hit)* cuando la contribución de la variable original es mayor al umbral.
4. El paso 3 se realiza iterativamente, donde la cantidad de iteraciones es establecida *a priori*. Si una variable no se ha registrado como un *hit* en cierta cantidad de iteraciones (15 veces, por ejemplo), esta es rechazada y eliminada de la matriz original.
5. El algoritmo se da por finalizado cuando, luego de una cantidad de iteraciones, todas las características han sido confirmadas o rechazadas. La cantidad de iteraciones totales del algoritmo también es establecida *a priori* por el analista.

Veamos un ejemplo ilustrativo con cinco filas y tres variables predictivas (edad, altura y peso). El resultado de la matriz X duplicada y con sus características sombra con valores «revueltos» se presenta en la figura 7.4.

Figura 7.4: Algoritmo BORUTA, paso 1
(construcción características sombra)

| | Edad | Altura | Peso | Sombra_Edad | Sombra_Altura | Sombra_Peso |
|---|---|---|---|---|---|---|
| **0** | 28 | 178 | 80 | 61 | 175 | 80 |
| **1** | 61 | 175 | 72 | 52 | 178 | 95 |
| **2** | 15 | 165 | 58 | 15 | 184 | 72 |
| **3** | 52 | 153 | 64 | 28 | 153 | 64 |
| **4** | 44 | 184 | 95 | 44 | 165 | 58 |

Una vez que se ejecuta RF sobre esta base, se obtiene la importancia de cada atributo. Se define como acierto o *hit* si se cumple:

Ecuación 7.11

$$\%\ Importancia\ de\ la\ variable\ original\ i > Max\ [\%\ Importancia\ de\ variables\ sombra\ Edad, Altura\ y\ Peso]$$

Cabe destacar que los valores en las características de sombra al estar ordenadas aleatoriamente, su importancia será menor en comparación con las originales. La importancia (en porcentaje) y el vector de aciertos se observa en la figura 7.5.

Figura 7.5: Algoritmo BORUTA, paso 3
(construcción vector de aciertos)

| | Edad | Altura | Peso | Sombra_Edad | Sombra_Altura | Sombra_Peso |
|---|---|---|---|---|---|---|
| **feature importance %** | 36 | 9 | 18 | 15 | 12 | 10 |
| **Acierto** | 1 | 0 | 1 | - | - | - |

Como se puede observar en la figura 7.5, solamente Edad y Peso están sobre el umbral, que en este caso es 15%, representado por Sombra_Edad. Entonces, ¿deberíamos dejar estas dos variables y eliminar Altura? Como es común en *machine learning*, se recomienda generar un proceso iterativo como se indica en el paso 4 de BORUTA. Por ejemplo, el resultado de repetir el procedimiento 20 veces se presenta en la figura 7.6.

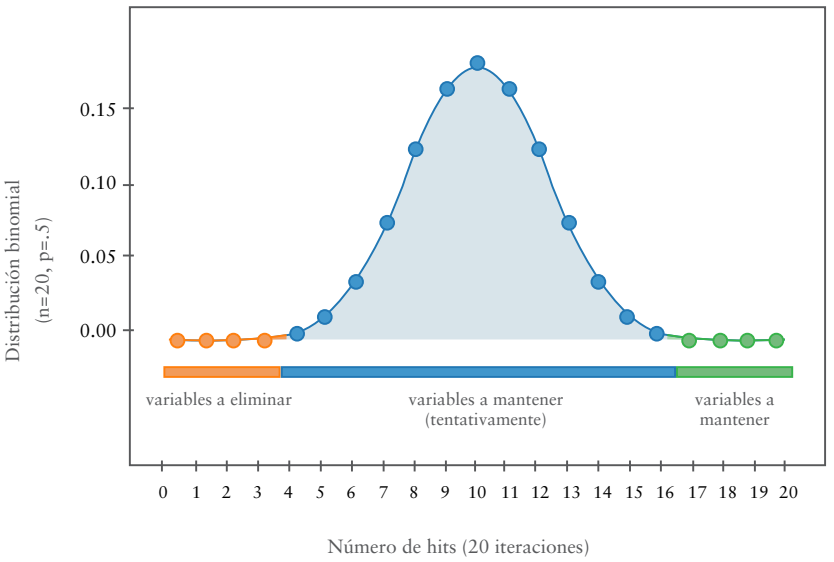
Figura 7.6: Algoritmo BORUTA, paso 4
(proceso iterativo de calcular *hits*)

| | Edad | Altura | Peso | |
|---|---|---|---|---|
| **Acierto** | 20 | 0 | 8 | |

De acuerdo con la figura 7.6, edad es acierto las 20 veces, mientras que altura ninguna vez. Parece bastante evidente que la Edad debería quedar, mientras que la Altura debería eliminarse. Pero ¿y el Peso? ¿Son suficiente 8 *hits*? Dado que cada experimento es independiente con una salida binaria (acierto o no), esta serie de experimentos sigue una distribución estadística conocida: la binomial.

La propiedad estadística anterior nos permite construir un gráfico como el de la figura 7.7, en donde se definen tres áreas: los atributos a eliminar (zona roja, aquellos con menos *hits*), los atributos a mantener (zona verde, aquellos con más *hits*) y una zona intermedia donde BORUTA no se «muestra confiado» en tomar una decisión, pero de forma conservadora sugiere mantenerlos.

Figura 7.7: Algoritmo BORUTA, paso 5
(decisión de eliminación de variables)



Como se observa en la figura 7.7, el área de rechazo se encuentra entre 0 y 3 aciertos, que es donde se encuentra Altura, luego el área de indecisión que es entre 4 y 16 aciertos, precisamente donde se encuentra Peso y, finalmente, el área de aceptación que es entre 17 y 20 aciertos, donde se encuentra Edad. En este ejemplo, cada cola representa 0.5% de la distribución, lo cual es una decisión del modelador.

En conclusión, en este ejemplo BORUTA nos ayudó a identificar las variables a utilizar dado que presentaron un mayor porcentaje de importancia en el conjunto de datos presentes: Edad y Peso, dejando fuera a la variable Altura.

### *7.3.2 Interpretando modelos de caja negra II: TREESHAP*

Una de las tareas que debe realizar como *data scientist* corresponde a la comunicación efectiva de la información que obtenga con los tomadores de decisiones, lo cual requiere poder explicar el fenómeno subyacente en sus datos de manera de generar confianza en los tomadores de decisión y cumplir requisitos regulatorios.

Para realizar la tarea de interpretación de estos modelos, surgen herramientas como TreeSHAP, que corresponde a un enfoque de teoría de juegos para explicar el resultado de modelos de árboles de decisión.

Similar a BORUTA, el enfoque en sí mismo no propone una selección de variable *a priori,* sino que se encarga de explicar la predicción para todo $\overline{X_i}$ de un conjunto de datos utilizado en un modelo de árbol como una suma de contribuciones de sus atributos evaluados para cada $\overline{X_i}$, lo cual permite *a posteriori* realizar una selección de los atributos que contribuyan en un grado significativo en la explicación o predicción del fenómeno en estudio.

A modo más general, el enfoque teórico de juegos SHAP o *SHapley Additive exPlanation* (explicación aditiva de Shapley) propone que los valores para las características individuales (variables independientes) están en un juego cooperativo cuyo «pago» *(payouts)* es la predicción ($Y_i$) dentro del contexto de un determinado modelo

predictivo. El desempeño de un modelo se desagrega en un conjunto de pagos que contribuyen al éxito de la predicción. El valor Shapley permite distribuir estos pagos entre los «jugadores», que son las variables del modelo.

Por lo tanto, el «juego» es la predicción de un ejemplo en la base de datos, los valores de las características individuales de la instancia $X_i$ corresponden a los jugadores. El valor Shapley corresponde al promedio de las contribuciones marginales en todas las combinaciones posibles para el atributo en estudio.

Dado que el problema de evaluar los posibles valores de un atributo en combinación con los otros y su impacto en la predicción es un problema combinatorial, calcular los valores SHAP con exactitud se vuelve un desafío computacional importante.

Para solucionar este problema se han diseñado diversas heurísticas (estrategias rápidas y eficientes que llevan a buenos resultados, pero que no garantizan que sea la solución óptima). Una de ellas, TREESHAP, es una estrategia diseñada para modelos con estructura de árbol.

La ventaja de la estrategia de árboles de decisión para el cálculo de la contribución marginal de cada atributo es que un árbol puede recibir todas las combinaciones posibles de atributos al mismo tiempo. La idea es llevar el registro del número de subconjuntos para cada nodo de decisión del árbol. Por ejemplo, si la primera ramificación del árbol se realiza con el atributo Edad, entonces todos los subconjuntos que contienen Edad irán por un camino, mientras que los que no contienen Edad irán por el otro camino y recibirán una contribución menor.

## 7.5 Extensión multiclase

Como se mencionó en la introducción de este capítulo, existen algunos desafíos importantes de la analítica de negocios en donde se deben clasificar tres o más clases, tales como en clasificación de riesgo. Este problema es más común en clasificación de imágenes, donde un objeto puede ser una clase entre varias posibles.

Una de las principales aplicaciones multiclase en negocios es (o, más bien, fue) el reconocimiento de dígitos escritos a mano para la lectura automática de cheques. Cada dígito escrito a mano es individualizado en una imagen única y utilizado para entrenar modelos multiclase. Cada dígito posible (del 0 al 9) es una clase a predecir. Este desafío fue impulsado por los laboratorios AT&T Bell en las décadas de los ochenta y noventa, junto a los principales investigadores en el área de inteligencia artificial (incluyendo a Yann Lecun, uno de los padres del aprendizaje profundo). Muchos de los métodos actuales de aprendizaje de máquinas, tales como las redes convolucionales (que veremos en el siguiente capítulo), fueron inventados con el fin de mejorar el desempeño predictivo para esta aplicación.

Todos los modelos de clasificación vistos anteriormente pueden ser extendidos a ajustes de multiclase. Centraremos este tópico en los modelos regresión logística y árbol de decisión, pero la lógica detrás de esta extensión se puede extrapolar a cualquier modelo.

## Enfoques Uno vs. Uno y Uno vs. Resto

Pensemos en la regresión logística, que construye una función para separar las dos clases. Su interpretación geométrica es la de un hiperplano, es decir, una recta en varias dimensiones. Cuando trabajamos con varias clases, lo que debemos hacer es construir varios hiperplanos separadores, para luego combinarlos.

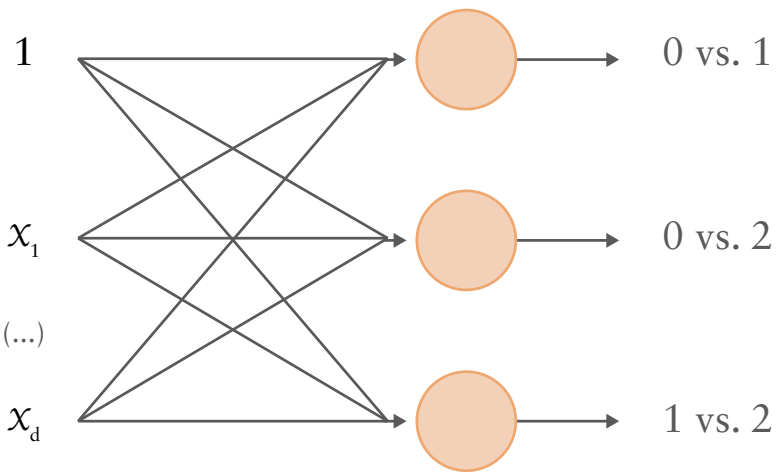
Pensemos en una situación donde queremos efectivamente saber el nivel de riesgo de crédito que presentan los clientes de una institución financiera, donde existen tres niveles: 0, 1 y 2. Un primer enfoque de construcción de hiperplanos es el llamado Uno versus Uno, que separa los datos en miniproblemas de clasificación binaria con dos clases. Es necesario evaluar toda la combinación de pares de clases, es decir, se construyen $\frac{K(K-1)}{2}$ hiperplanos cuando se tienen K clases.

Como hemos mencionado anteriormente, los problemas combinatoriales son costosos en lo computacional. Lo bueno es que el número K rara vez es muy grande, mientras que los problemas

resultantes tampoco son tan grandes, ya que solo consideramos los datos asociados a las clases respectivas (Baesens, 2014).

Una vez que todos los clasificadores fueron construidos, el resultado final se obtiene mediante una votación de la clase que fue más votada en la estimación. En otras palabras, uno evalúa un nuevo ejemplo con cada uno de los hiperplanos e identifica a qué lado cae. Esto se registra como un «voto» para esa clase. Luego se cuentan todos los votos y se asigna el ejemplo a la clase más votada. Este enfoque se presenta en la figura 7.8.

Figura 7.8: Enfoque «Uno vs. Uno» para clasificación multiclase

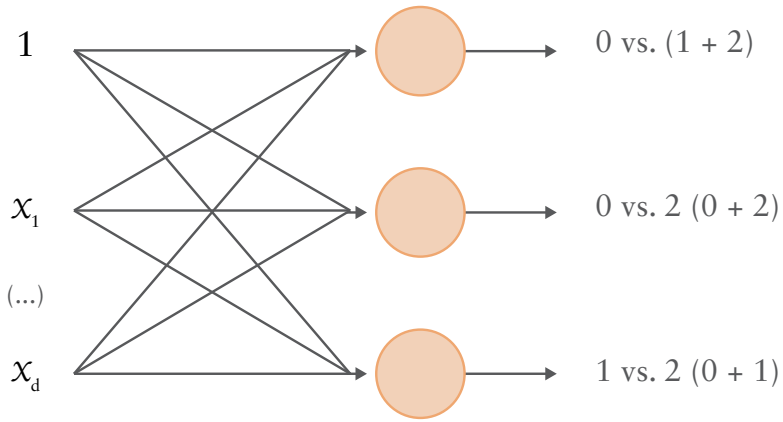



Por otro lado, otra forma de ver el problema multiclase es con un enfoque «Uno vs. Resto», donde se construye un hiperplano separador para cada clase «compitiendo» contra todas las otras clases fusionadas como una categoría única. En este caso, solo se construyen K funciones, por lo que puede ser más eficiente computacionalmente que Uno vs. Uno cuando se tienen muchas clases. Sin embargo, cada problema resultante es más lento en «Uno vs. Resto», ya que contienen todos los datos (López *et al.*, 2017).

Una vez que todos los hiperplanos fueron construidos, el resultado final se obtiene seleccionando la clase que tiene una mayor probabilidad *a posteriori*. En otras palabras, uno evalúa un nuevo ejemplo con cada uno de los hiperplanos y calcula la probabilidad de

pertenecer a la clase única. El ejemplo se asigna entonces a la clase con mayor probabilidad de las K posibles. Este enfoque se presenta en la figura 7.9.

Figura 7.9: Enfoque «Uno vs. Resto» para clasificación multiclase




## Enfoques multiclase simultáneos

Existen además estrategias multiclase que pueden resolver el problema con un modelo único. Algunos, como las redes neuronales (discutidas en el capítulo siguiente) o k vecinos más cercanos, están diseñados para manejar varias clases. En k-NN, se construye un vecindario para cada nuevo objeto, considerando los K elementos del conjunto de entrenamiento más parecidos a este. Una vez definido el vecindario, se toma la moda de las K etiquetas de los objetos del vecindario. Este proceso no cambia nada si tenemos en cuenta más de dos clases.

El árbol de decisión también puede ser extendido a multiclase de manera sencilla, dado que el procedimiento es general a la cantidad de clases: se elige el atributo que más logra reducir el «desorden» en el sistema, se ramifica con este atributo y se vuelve a intentar ramificar cada nodo resultante de la misma forma. Cada nodo y cada hoja tendrán ahora elementos de varias clases. Para definir la etiqueta del nodo hoja se toma la moda, al igual que en el caso de dos clases.

Lo importante es considerar medidas de impureza del nodo que involucren varias clases, por lo que métricas como entropía o Gini deben ser ajustadas. Este ajuste es bastante simple:

Ecuación 7.13

$$Entropy(S) = -\sum_{k=1}^{K} p_k log_2(p_k)$$

Ecuación 7.12

$$Gini(S) = \sum_{k=1}^{K} p_k(1 - p_k)$$

Es decir, ahora las sumatorias tienen más elementos, pero son, en esencia, las mismas fórmulas. Para modelos basados en árboles, tales como RF, la adaptación es simple: se construyen los árboles multiclase de la misma forma aleatoria y, al final, se realiza la votación por mayoría. La única diferencia es que hay más de dos clases al momento de obtener la moda de las etiquetas.

## Referencias


Baesens, B. (2014). *Analytics in a* Big Data *World*. New Jersey: John Wiley and Sons.

Baesens, B., Van Vlasselaer, V & Verbeke, W. (2015). Fraud *Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*. New Jersey: John Wiley and Sons.

Baesens, B., Roesch, D. & Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. New Jersey: John Wiley and Sons.

Bauer, E. & Kohavi, R. (1999). «An empirical comparison of voting classification algorithms: Bagging, boosting, and variants». *Machine Learning,* 36, 105-139.

Breiman, L. (2001). «Random Forests». *Machine Learning,* 45, 5-32.

Han, J., Kamber, M. & Pei, J. (2011). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

López, J., Maldonado, S. & Carrasco, M. (2017). «A robust formulation for twin multiclass support vector machine». *Applied Intelligence*, 47(4), 1031-1043.

Verbeke, W., Baesens, B. & Bravo, C. (2017). *Profit Driven Business Analytics*. New Jersey: Wiley.

# Aplicación 4: Experiencia del consumidor (CX)

El crecimiento exponencial de la inteligencia artificial (IA) está cambiando radicalmente la experiencia del consumidor (*Customer eXperience,* o CX). En marzo de 2020, MIT *Technology Review Insights* presentó los resultados de una encuesta realizada a mil líderes empresariales. Esta encuesta indica que el área de servicio al cliente es aquella donde más se usa IA (73%), seguido por los departamentos de ventas y marketing. Esta misma encuesta proyectó que el departamento de servicio al cliente será el área líder en el uso de IA para el año 2022 (McCauley, 2020).

De acuerdo con McAfee *et al.* (2012), las organizaciones que logran administrar adecuadamente la retroalimentación de los consumidores son, en promedio, un 6% más rentables y un 5% más productivas que otras empresas. Dentro de las estrategias utilizadas para identificar patrones en encuestas y comentarios de clientes, la IA es una de las que más adeptos está ganando dados los excelentes resultados que se están reportando en análisis de texto *(text analytics)* y procesamiento de lenguaje natural (*Natural Language Processing* o NLP).

Uno de los principales desafíos para entender al consumidor es ser capaz de analizar automáticamente respuestas a preguntas abiertas. Encuestas de selección múltiple son bastante fáciles de procesar y generar estadística descriptiva, sin embargo, son incapaces de proporcionar una visión completa de las percepciones de los clientes. Tampoco permiten descubrir aspectos nuevos asociados a su experiencia. En otras palabras, se limita al consumidor en ciertas alternativas, en vez de ofrecerle la oportunidad de explayarse libremente sobre su satisfacción con la empresa.

Este capítulo ofrece una discusión sobre los factores o *drivers* que afectan la satisfacción del cliente y propone una estrategia basada en el proceso CRISP-DM (ver capítulo 1) y en el modelo BERT para la extracción automática de conocimiento en encuestas de experiencia del consumidor. En particular, se estudia la estrategia *Net Promoter Score* (NPS) (Reicheld y Markey, 2011), una forma muy popular de recopilar información del consumidor.

## 1. Fundamentos de la experiencia y satisfacción del consumidor

La investigación científica en marketing y servicios ha dedicado bastantes estudios a las causas y consecuencias de la satisfacción del cliente (*customer satisfaction*, o CSAT), definida como «el juicio que realiza el consumidor a un producto o servicio con relación a si alcanza o no sus expectativas» (Gupta & Zeithaml, 2006). CSAT es un concepto más bien holístico y subjetivo, ya que refleja la percepción de un individuo sobre toda su experiencia con una empresa.

Existen diferentes herramientas desarrolladas por académicos para evaluar la satisfacción del cliente. Sin embargo, las empresas prefieren métodos y medidas simples debido a que encuestas largas tienden a abrumar a los participantes. Esto conlleva a tasas de respuesta bajas y sesgos.

Una de las medidas más populares es el NPS. Esta medida consiste en un puntaje de 0 a 10 asignado por el cliente a la siguiente pregunta: ¿con qué probabilidad recomendarías el producto/servicio a un amigo o compañero? Esta pregunta es seguida por una invitación a justificar su respuesta, con la siguiente pregunta abierta: ¿por qué? Los que puntúan más alto (9 o 10) se conocen como *promotores,* mientras que los que puntúan más bajo (0 a 6) se consideran *detractores*. En el medio se ubican los pasivos o indiferentes. Para una empresa es clave identificar los factores que influyen tanto a los promotores, que tienen una alta probabilidad de recomendar el producto o servicio, como a los detractores, que se mostrarán abiertamente críticos a la empresa.

Por otro lado, existen diversos estudios científicos en el área de calidad de servicios que proponen factores que explican los indicadores de satisfacción de los clientes. El modelo SERVQUAL fue el primero en medir la calidad de servicio. Incluye las siguientes dimensiones:

- **Confiabilidad:** capacidad de otorgar el servicio prometido de forma precisa y oportuna.
- **Responsabilidad:** capacidad de responder de forma rápida a requerimientos.
- **Garantía:** conocimiento y cortesía de los empleados, junto a su capacidad de transmitir confianza.
- **Empatía:** preocupación y atención individualizada ofrecida a los clientes, siendo capaces de reconocer sus emociones.

Diferentes investigadores han complementado con nuevos factores el modelo SERVQUAL, tales como:

- **Atmósfera:** los consumidores evalúan el diseño y ambiente de las tiendas en busca de señales, ya que estas ofrecen información confiable sobre el producto o servicio.
- **Variedad:** la amplitud en el surtido de productos o servicios ofrecidos, y su exclusividad.
- **Marca:** la capacidad que tiene el nombre de la marca de influenciar la percepción del cliente.
- **Conveniencia:** la capacidad de ofrecer consistencia, alternativas y puntualidad a nivel de canal.
- **Facilidad de uso:** qué tan fácil es hacer negocios con la compañía o, en otras palabras, el esfuerzo que requiere invertir el consumidor para interactuar con ella.
- **CX pasado:** todas las interacciones entre el cliente y la organización durante el ciclo de vida del cliente.
- **Precio:** las estrategias de precio juegan un rol clave en CX, ya sea como simple propuesta de valor o como estrategias de marketing adicionales (promociones o programas de lealtad).

## 2. Inteligencia artificial aplicada a CX

La minería de texto o *text analytics* es una disciplina que considera diversas técnicas descriptivas y predictivas para extraer conocimiento de texto de forma automática. Este proceso trae muchas ventajas a las empresas en términos de escalabilidad, precisión y confiabilidad, evitando posibles sesgos e ideas preconcebidas sobre los datos. Se estima que aproximadamente el 80% de la información corporativa viene en forma de texto (Ur-Rahman and Harding, 2012), por lo que representa un enorme potencial para encontrar patrones valiosos.

Dentro de las tareas de *text analytics* más consideradas en las empresas se encuentran la categorización de texto, la segmentación, el resumen de documentos y el análisis de sentimientos. La investigación en la retroalimentación de los clientes se ha enfocado en esta última tarea, que corresponde a encontrar patrones relacionados con opiniones y las emociones con que se expresan.

Las empresas suelen disponer de diversos recursos basados en texto, tales como comentarios, reclamos, sitios web, libros, emails y artículos. Pese a la diversidad de fuentes, los comentarios de los clientes son probablemente los recursos más valiosos. Debido a esto, son también los más estudiados. Algunas aplicaciones en esta línea son *reviews* sobre hoteles, automóviles o restaurantes (Vairetti *et al.*, 2020). También existen trabajos importantes en la línea de categorización de reclamos.

Uno de los principales desafíos en el área de CSAT es cómo proceder con la retroalimentación de los consumidores en base a preguntas abiertas. La extracción de conocimiento no es simple, ya que es difícil «guiar» el proceso de aprendizaje cuando no hay etiquetas ni claridad sobre las razones que justifican la evaluación. En este sentido, la relación entre el puntaje asignado por clientes bajo el enfoque NPS y los factores que explican la satisfacción del cliente es clave para tener una visión global sobre las percepciones de los clientes y poder tomar mejores decisiones.

Con el fin de hacernos cargo de este desafío, recientemente publicamos una estrategia de extracción de patrones para datos tipo

NPS utilizando técnicas descriptivas de *text analytics* y DL. Este modelo se presenta a continuación.

## 3. Propuesta para el análisis de datos de CX con IA

Con el objetivo de ganar *insights* valiosos sobre los factores que se asocian a CSAT, proponemos una estrategia de cuatro etapas que permita analizar tanto la puntuación NPS como las respuestas a la pregunta abierta. En particular, se buscan en el texto los 11 factores indicados anteriormente utilizando el método BERT. Este estudio es uno de los primeros en interpretar retroalimentación de los consumidores con base en la clasificación de texto.

La metodología propuesta es una adaptación del proceso CRISP-DM formalizado en el capítulo 1, que incluye las seis etapas ya discutidas en el libro: comprensión del negocio, comprensión de los datos, preparación de los datos, modelamiento, evaluación e implementación.

Como caso de estudio, la metodología fue aplicada a una base de datos de NPS consistente en 25,943 respuestas de consumidores asociadas a 39 compañías chilenas en 13 sectores industriales y servicios diferentes.

### *3.1 Etapa 1: Recolección, visualización y procesamiento de datos*

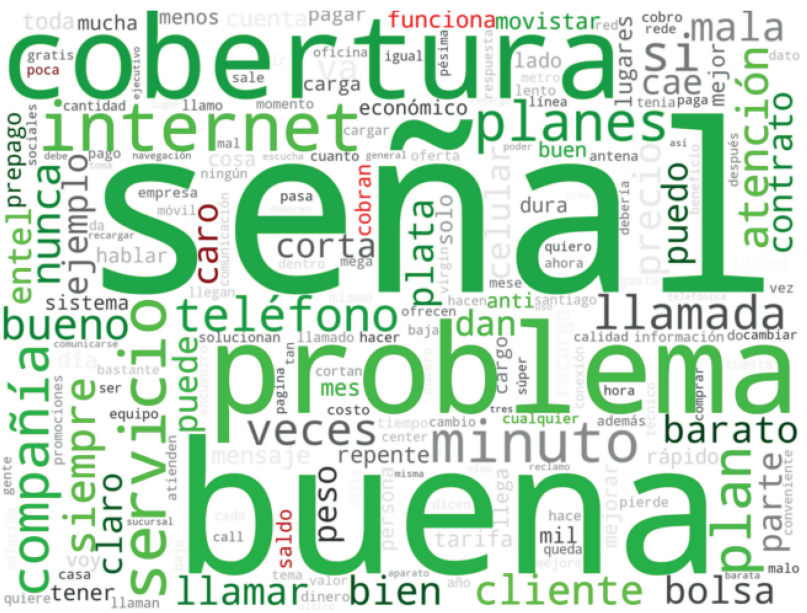
Esta etapa puede subdividirse, a su vez, en tres pasos: definición de los objetivos del minado de texto, visualización y estadística descriptiva, y preparación y procesamiento de datos. Para el caso de estudio se propone como objetivo del minado de texto la extracción automática de los 11 factores o dimensiones que explican la experiencia del consumidor y su relación con el puntaje NPS. El objetivo se facilita gracias a que se cuenta con una base de datos donde se etiquetó de forma manual cada respuesta abierta de los consumidores, identificando la presencia de hasta tres factores en

cada comentario. De esta forma es posible aplicar técnicas supervisadas para la categorización de texto.

Para el segundo paso se considera, adicional a las técnicas tradicionales de analítica descriptiva, la construcción de nubes de palabras *(word clouds)*, en donde coloreamos aquellas palabras y frases relevantes en base a la etiqueta predominante del puntaje NPS (promotor, pasivo o detractor).

Un ejemplo de estas nubes se presenta en la figura App4.1 para la industria del *retail*. En este caso consideramos palabras individuales (unigramas) para reflejar la importancia de un concepto. El tamaño de letra representa frecuencia (cuanto más grande, más frecuente), mientras que el color indica el promedio del puntaje NPS (redondeado a cero decimales) cuando un concepto es mencionado: verde para puntaje promedio 9 o superior (promotores), rojo si el puntaje promedio NPS es menor o igual a 6 (detractores), y gris en caso contrario (pasivos).

Figura App4.1: *Word clouds*



La figura App4.1 destaca conceptos como «precio» (en verde) o «caro» (en rojo), que claramente se asocian al factor precio, y otros como «variedad», «atención» o «tienda», que se asocian a otros factores que afectan la calidad de servicio. Esto confirma la relación directa que existe entre las respuestas abiertas de los consumidores y los factores de CSAT.

El *output* de este segundo paso es un reporte de la calidad de los datos y del proceso de exploración, el cual incluye los primeros patrones y define las directrices para un preprocesamiento de datos adecuado, que es justamente el tercer paso.

La preparación de los datos incluye selección, etiquetado y limpieza. En nuestro caso de estudio, los datos fueron recolectados de forma telefónica y transcritos manualmente. En el mismo proceso, los datos fueron etiquetados. Para ilustrar este proceso, veamos los siguientes dos ejemplos para la industria de telefonía móvil:

- El comentario «tiene tarifas convenientes y se puede hablar más por menos dinero» se relaciona con el factor «precio».
- La oración «la calidad de la señal es terrible y las funcionalidades del teléfono son limitadas» se asocian al factor «confiabilidad».

La siguiente tabla presenta la estadística descriptiva de las bases finales (una por industria), indicando el número de etiquetas (factores) incluidas en los comentarios, el número de comentarios (respuestas abiertas), la razón entre etiquetas y comentarios, la fracción de promotores y detractores (en porcentaje) y el puntaje NPS, calculado como el porcentaje de promotores menos el porcentaje de detractores.

Tabla App4.1

| Industria | Etiquetas | Comentarios | Eti. / Com. | Promotores | Detractores | NPS |
|---|---|---|---|---|---|---|
| Gas | 2849 | 1477 | 1.93 | 71% | 10% | 61 |
| Clínicas | 3031 | 1347 | 2.25 | 52% | 16% | 36 |
| Gasolineras | 2871 | 1466 | 1.96 | 51% | 17% | 34 |
| Tarjetas *retail* | 3907 | 2075 | 1.88 | 50% | 25% | 25 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Bancos | 4875 | 2315 | 2.11 | 43% | 25% | 18 |
| *Retail* | 4116 | 1989 | 2.07 | 40% | 27% | 13 |
| Teléfono móvil | 8327 | 4214 | 1.98 | 43% | 30% | 13 |
| Supermercados | 4095 | 1992 | 2.06 | 37% | 29% | 8 |
| Televisor | 3789 | 1983 | 1.91 | 36% | 34% | 2 |
| Farmacias | 2862 | 1490 | 1.92 | 33% | 37% | -4 |
| Seguro de salud | 3994 | 1954 | 2.04 | 27% | 39% | -12 |
| Fondo pensiones | 4030 | 2121 | 1.9 | 26% | 41% | -15 |
| Internet | 2233 | 1155 | 1.93 | 28% | 44% | -16 |
| **Promedio** | **3921** | **1968** | **2.00** | **41.3%** | **28.8%** | **13** |

En esta tabla se observa que las bases son de tamaño relativamente similar en términos de etiquetas y comentarios. En promedio, cada participante de la encuesta menciona dos factores en sus respuestas. En contraste, la distribución de los puntajes NPS es bastante asimétrica entre industrias, mostrando niveles de satisfacción muy diferentes: la mejor industria es Gas Natural (61%), mientras que la peor es Internet (Cable/Fibra óptica, -16%). Este resultado confirma la importancia de construir modelos personalizados para cada industria.

Con relación a las etapas de preprocesamiento específicas para *text analytics*, seguimos procedimientos estándar de eliminación de *stopwords* (palabras frecuentes, como artículos y pronombres) y *stemming* (método para reducir la cantidad de palabras diferentes que consiste en llevar palabras derivadas a forma raíz).

## 3.2 Etapa 2: Word Embedding

El texto es un tipo de datos no estructurado que requiere de transformaciones para darle sentido matemático. Este proceso, conocido como *word embedding*, se ha realizado de manera tradicional con el enfoque TF/IDF. Esta estrategia construye vectores

utilizando palabras únicas o bigramas de documentos, de tal forma que mientras más aparece una palabra o concepto, mayor es su contribución en el modelo. Estos vectores alimentan luego un modelo de clasificación tradicional.

Enfoques más recientes, sin embargo, permiten analizar oraciones completas con el fin de interpretar el sentido de cada palabra, empotrando el proceso de *embedding* dentro del algoritmo de aprendizaje. El modelo BERT *(Bidirectional Encoder Representations from Transformer)* de Google (Devlin *et al.*, 2018) ha ganado popularidad debido a su capacidad predictiva muy superior al enfoque tradicional. Su característica clave es su capacidad de «leer» una secuencia entera de palabras, interpretando el contexto de una palabra en base a los conceptos adyacentes.

BERT es un *transformer,* es decir, tiene una estructura de dos mecanismos: un codificador que lee los datos de entrada y un decodificador que realiza la predicción. BERT enmascara un 15% de las palabras de la secuencia de entrada de forma aleatoria y busca predecir las palabras faltantes con el decodificador. BERT también sigue una estrategia similar para aprender relaciones entre oraciones: recibe pares de oraciones como entrada y busca predecir si la segunda oración es subsecuente de la primera. Para ello, 50% de las entradas son dos oraciones consecutivas, mientras que el otro 50% son oraciones no consecutivas seleccionadas al azar.

Otra característica clave de BERT es su habilidad para combinar recursos de otras fuentes de texto. BERT preentrena un modelo con texto de diferentes fuentes, tales como Wikipedia o libros, usando redes neuronales profundas. Para ello requiere de millones de oraciones para poder hacer sentido, lo que puede no ser computacionalmente factible de realizar para usuarios regulares. Sin embargo, la ventaja es que podemos tomar un modelo preentrenado de Google (los creadores de BERT), u otro proyecto abierto al público general, y solo adaptarlo a nuestros datos. Este proceso se conoce como *fine-tuning* y permite customizar un modelo general a los datos disponibles sin perder la comprensión del lenguaje obtenida con el modelo preentrenado (Devlin *et al.*, 2018). Esto es particularmente

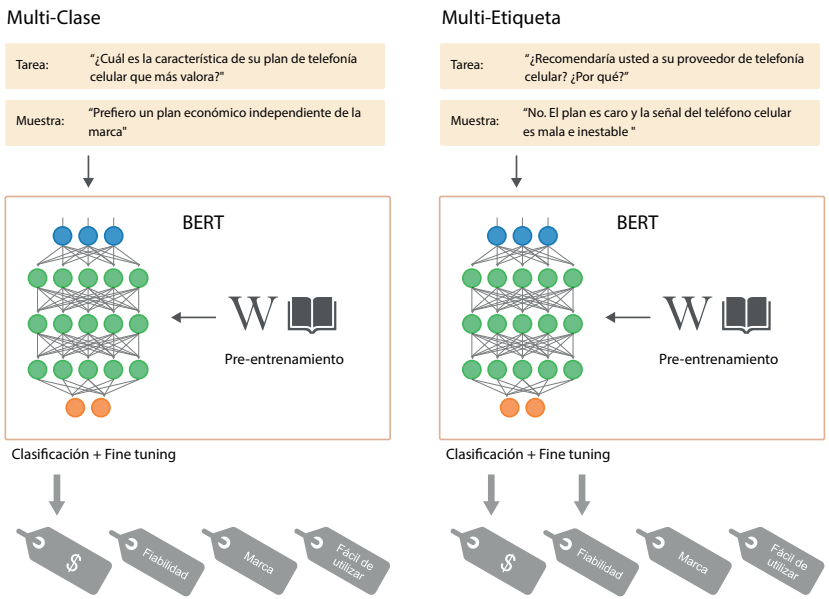
útil cuando se cuenta con bases de datos relativamente pequeñas, como las utilizadas en este estudio.

## 3.3 Etapa 3: Clasificación multietiqueta

Clasificación es probablemente la tarea más común en el mundo del aprendizaje de máquinas. Su objetivo es predecir a qué clase pertenece una observación, extrapolando las características de ejemplos pasados etiquetados en dos (clasificación binaria) o más clases (clasificación multiclase). En esta última tarea, cada ejemplo (en nuestro caso, comentarios) pertenece a una y solo una clase. Sin embargo, un caso especial es la clasificación multietiqueta, donde cada comentario puede tener una o más clases. En este estudio, por ejemplo, una respuesta de un consumidor puede incluir varios factores CSAT, refiriéndose tanto al precio del producto o servicio como a la calidad de servicio vivida.

La figura App4.2 ilustra tanto el funcionamiento de BERT (etapas de preentrenamiento y *fine-tuning*) como la diferencia entre un modelo multiclase y uno multietiqueta. Si, por ejemplo, se le pide a un participante que elija solo una alternativa en su comentario, el problema será multiclase (ver lado izquierdo de la figura App4.2). En cambio, una respuesta abierta donde el usuario pueda explayarse libremente sobre su experiencia dentro de un mismo comentario se vuelve un problema multietiqueta (ver lado derecho de la figura App4.2).

Figura App4.2: Diferencia entre enfoques multiclase Multi-label




Utilizando un modelo preentrenado como punto de partida, podemos ajustar los pesos de la red con una capa de salida *ad-hoc* para la tarea multietiqueta, mientras alimentamos con los datos nuevos. De esta manera el modelo no solo construye el clasificador que comete menos errores, sino también la mejor representación matemática de nuestros datos, propagando los errores a través del *transformer*. Este proceso también se conoce como *transfer learning*, ya que el conocimiento adquirido por una máquina se transfiere de una tarea a otra.

La capa de salida propuesta es una estándar para la tarea de multiclase o multietiqueta, que utiliza *cross-entropy* como función de pérdida y *softmax* como función de activación (combinación de varias funciones sigmoidales para diferentes clases).

## 3.4 Etapa 4: Predicción e interpretación de patrones

La etapa final consiste en la validación del modelo y evaluación de resultados. Para esto, sugerimos la estrategia de dividir el conjunto

de datos en tres subconjuntos: entrenamiento, validación y testeo. Los conjuntos de entrenamiento y validación (60% y 10% de los datos, respectivamente) se utilizan para construir el modelo y elegir los parámetros, mientras que el conjunto de testeo (30% restante) se utiliza para la evaluación final y permanece oculto del proceso de aprendizaje.

Para evaluar el modelo, se considera la medida *F1* ponderada, la cual es utilizada frecuentemente en tareas multietiqueta. Para una clase (factor) *j*, la precisión $P_j$ corresponde a todos los casos correctamente clasificados como esta clase *(true positives)* divididos por todos elementos predichos como esta clase *(true positives+false positives)*, mientras que *recall* o exhaustividad es el total de casos correctamente clasificados como esta clase *(true positives)* dividido por todos elementos de esta clase *(true positives+false negatives)*. El F1 de la clase *j* es entonces la media armónica entre precisión y *recall:*

Ecuación App4.1

$$F1_j = 2P_jR_j / (P_j + R_j)$$

Finalmente, la medida *F1* ponderada se calcula como la suma de todos los $F1_j$ ponderados por la fracción de casos que pertenecen a esta clase.

Comparamos el desempeño de BERT con dos modelos alternativos: el enfoque TF/IDF tradicional alimentando una red neuronal tradicional (ANN, una capa oculta) y una SVM. BERT obtuvo un *F1* de 0.72, mientras que ANN y SVM obtuvieron solo 0.42 y 0.37, respectivamente. Este resultado es muy potente ya que, por un lado, demuestra el excelente desempeño de BERT. Es importante destacar que 0.72 es un excelente desempeño para un modelo multietiqueta de 11 clases, por lo que demuestra el éxito del modelo. Por otro lado, los resultados de ANN y SVM son más bien regulares, por lo que se confirma que nuestra estrategia de categorización automática de comentarios solo es viable utilizando las últimas tecnologías de aprendizaje profundo.

Finalmente, la tabla App4.2 presenta el desempeño predictivo en términos de F1 desagregado a nivel de factor y sector industrial. En esta tabla es posible observar que el mejor desempeño ocurre para el factor «precio», seguido por «responsabilidad». Era esperable ya que estos factores se pueden ligar directamente a palabras como «barato», «caro», «precio» o «respuesta». Sin embargo, otros factores como «conveniencia» o «CX pasada» son más abstractos y, por ende, el modelo no es capaz de aprenderlos con la misma precisión.

La tabla App4.2 muestra también diferencias importantes entre industrias en términos de capacidad predictiva. Algunos valores de F1 caen incluso a cero en algunas industrias, siendo clasificados de forma casi perfecta en otras. Esto se debe principalmente a la disponibilidad de datos, ya que en algunas industrias muchos factores CSAT están subrepresentados, provocando un desempeño desigual.

Tabla App4.2

| Industria | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Gas | 0.56 | 0.54 | 0.17 | 0.59 | 0.37 | 0.48 | 0.27 | 0.56 | 0.75 | 0.84 | 0.87 |
| Clínicas | 0.63 | 0.80 | 0.74 | 0.48 | 0.00 | 0.59 | 0.52 | 0.00 | 0.85 | 0.68 | 0.71 |
| Gasolineras | 0.68 | 0.57 | 0.79 | 0.51 | 0.00 | 0.59 | 0.71 | 0.67 | 0.84 | 0.08 | 0.68 |
| Tarjetas *retail* | 0.69 | 0.15 | 0.00 | 0.38 | 0.39 | 0.67 | 0.13 | 0.73 | 0.90 | 0.71 | 0.60 |
| Bancos | 0.36 | 0.76 | 0.71 | 0.61 | 0.33 | 0.54 | 0.57 | 0.52 | 0.75 | 0.61 | 0.69 |
| *Retail* | 0.88 | 0.75 | 0.70 | 0.47 | 0.50 | 0.46 | 0.67 | 0.00 | 0.78 | 0.48 | 0.65 |
| Teléfono móvil | 0.56 | 0.44 | 0.38 | 0.29 | 0.26 | 0.34 | 0.24 | 0.49 | 0.78 | 0.92 | 0.77 |
| Supermercados | 0.89 | 0.64 | 0.73 | 0.26 | 0.00 | 0.00 | 0.59 | 0.00 | 0.82 | 0.46 | 0.73 |
| Televisor | 0.36 | 0.00 | 0.85 | 0.29 | 0.71 | 0.29 | 0.29 | 0.52 | 0.77 | 0.84 | 0.81 |
| Farmacias | 0.85 | 0.67 | 0.73 | 0.61 | 0.00 | 0.00 | 0.72 | 0.00 | 0.85 | 0.20 | 0.84 |
| Seguro de salud | 0.78 | 0.44 | 0.48 | 0.45 | 0.42 | 0.58 | 0.52 | 0.26 | 0.83 | 0.57 | 0.71 |
| Fondo pensiones | 0.38 | 0.51 | 0.86 | 0.71 | 0.00 | 0.65 | 0.65 | 0.81 | 0.81 | 0.85 | 0.74 |
| Internet | 0.44 | 0.00 | 0.00 | 0.33 | 0.00 | 0.38 | 0.00 | 0.46 | 0.67 | 0.92 | 0.78 |
| **Promedio** | **0.62** | **0.48** | **0.55** | **0.46** | **0.23** | **0.43** | **0.45** | **0.39** | **0.80** | **0.63** | **0.74** |

1. Variedad
2. Garantía
3. Atmósfera
4. Marca
5. Conveniencia
6. Facilidad de uso
7. Empatía
8. CX pasado
9. Precio
10. Confiabilidad
11. Responsabilidad

Nuestra recomendación principal es desarrollar modelos a nivel de industria en vez de un modelo único con el fin de modelar mejor las complejidades de cada sector, ya que las palabras y los conceptos relevantes claramente varían de industria en industria. Por ejemplo, la palabra «señal» asociada a un teléfono móvil es clave para entender la experiencia del consumidor en la industria de telefonía, ya que define confiabilidad en telecomunicaciones. Sin embargo, este concepto es irrelevante en otras industrias.

## Referencias

Devlin, J., Chang, M. -W., Lee, K. & Toutanova, K. (2018). Bert: *Pretraining of deep bidirectional transformers for language understanding*. ArXiv.cs.CL 1810.04805.

Gupta, S. & Zeithaml, V. (2006). «Customer metrics and their impact on financial performance». *Marketing science*, 25(6), 718-739.

Maldonado, S., López, J. & Vairetti, C. (2020). «Profit-based Churn Prediction based on Minimax Probability Machines». *European Journal of Operational Research,* 284(1), 273-284.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. & Barton, D. (2012). «Big data: the management revolution». *Harvard business review*, 90(10), 60-68.

McCauley, D. (2020). «The global AI agenda: Promise, reality, and a future of data sharing». MIT *Technology Review Insights,* https://www.technologyreview.com/2020/03/26/950287/the-global-ai-agenda-promise-reality-and-a-future-of-data-sharing/

Reichheld, F. F. & Markey, R. (2011). *The ultimate question 2.0: How net promoter companies thrive in a customer-driven world*. New York: Gildan Media.

Ur-Rahman, N. & Harding, J. A. (2012). «Textual data mining for industrial knowledge management and text classification: A business oriented approach». *Expert Systems with Applications,* 39(5), 4729-4739.

# 8. *Predictive Analytics* III: Inteligencia artificial

Además de los modelos transparentes y de tipo «caja negra» presentados en los capítulos anteriores, existe otra clase de modelos predictivos propios del área de inteligencia artificial. Si bien este tipo de modelos también se considera como de «caja negra», hemos decidido dedicarle un capítulo exclusivo debido a su importancia y reciente éxito.

En este capítulo se introducirán los modelos de redes neuronales y los conceptos más importantes para crear modelos *Deep Learning* (DL), en conjunto con una guía práctica para la construcción y validación de estos modelos.

## 8.1 Definición de una red neuronal

A finales de la década de los cincuenta, el psicólogo estadounidense Frank Rosenblatt propuso el primer algoritmo de redes neuronales moderno, el cual simula el mecanismo de aprendizaje en organismos biológicos.

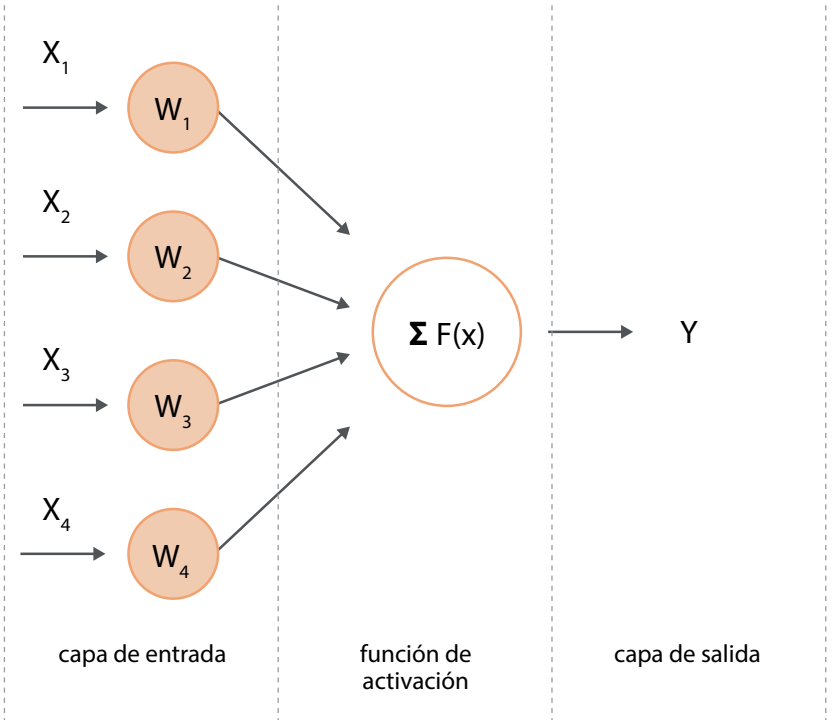
El paralelo que estableció fue el siguiente: el proceso de aprendizaje de una red neuronal artificial (*Artificial Neural Network* o ANN) es muy similar al que ocurre con las células encargadas de las funciones del sistema nervioso, llamadas **neuronas**. Estas últimas se encuentran conectadas entre sí, formando regiones de conexión, denominadas **sinapsis**.

Uno de los aspectos comunes más importantes para ambos tipos de redes es que ambas aprenden en respuesta a la recepción de estímulos externos, reflejados en la sinapsis de las neuronas. A

este cambio se le conoce, en los organismos vivos, como **aprendizaje** (Rosenblatt,1958).

Así, surge la forma más simple de una red neuronal llamada Perceptrón, la cual permite clasificar patrones linealmente separables, es decir, separar perfectamente las observaciones con un hiperplano (un modelo lineal, como el de la regresión logística) con la finalidad de clasificar nuevos ejemplos.

Figura 8.1: Arquitectura del modelo Perceptrón




En la figura 8.1, se puede observar el modelo del Perceptrón propuesto por Rosenblatt y sus principales componentes (Han *et al.* 2011):

- **Nodos de entrada:** *inputs* para una determinada capa, que reciben un valor numérico. Los *inputs* iniciales corresponden a las variables de entrada del modelo (matriz X).

- **Pesos:** ponderación o importancia de cada neurona en la red, también es un valor numérico. En el modelo Perceptrón equivalen a los coeficientes que definen las pendientes del hiperplano, que se representaban con la letra griega beta en la regresión logística.
- **Nodos de salida:** *outputs* de una determinada capa, corresponden a un valor numérico. El *output* final del modelo corresponde a la predicción.
- **Capa:** conjunto de conexiones entre nodos de entrada y de salida, que considera determinados pesos para cada nodo de entrada y una función de activación para la salida.
- **Función de activación:** función aplicada pos o previo a la ponderación de los nodos de entrada. En la figura 8.1 se muestra una función de suma, que resulta en la combinación de las variables originales (capa de entrada) en forma de una suma ponderada. En clasificación binaria es común considerar, además, una función de activación sigmoidal, similar a la función de enlace *logit* descrita para la regresión logística.
- **Función de pérdida:** además de la función de activación, también existe una función de pérdida para cuantificar el error, o diferencia, entre el valor real (vector de etiquetas Y) y el valor predicho (*output* final). La función de pérdida está relacionada con la naturaleza del problema (clasificación, regresión u otro).

Como se puede deducir de las definiciones, existe una relación muy cercana entre el modelo Perceptrón y la regresión logística. Consideremos un modelo de cuatro variables:

Ecuación 8.1

$$P_{y=1} = \frac{1}{(1 + e^{-(\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\beta_4X_4)})}$$

Donde $P_{y=1}$ sería el único nodo de salida, la función de activación es la sigmoidal $(e^{-(\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\beta_4X_4)})$, los nodos de entrada los valores de las variables $X_i : (i \in 1,2,3,4)$, y $\beta_1 : (i \in 1,2,3,4)$

representan los pesos, que en la notación de redes neuronales se representan con la letra «w» $(w_i \in 1,2,3,4)$.

El modelo Perceptrón, sin embargo, tuvo escaso éxito. A principios de la década de los setenta, los reconocidos académicos estadounidenses Minsky y Marvin argumentaron en contra de la factibilidad de entrenar este modelo ya que, desde el punto matemático y de recursos para computar las operaciones matemáticas necesarias, existían limitantes importantes al respecto (Minsky and Papert, 2017).

Así, las redes neuronales quedaron prácticamente en desuso hasta los años ochenta, para retomar fuerza con la aparición del algoritmo *backpropagation,* la base para encontrar la solución al problema de minimizar la función de pérdida. Este algoritmo se formaliza en la siguiente sección.

Adicionalmente, por esos años el modelo Perceptrón fue reemplazado por un modelo multicapa llamado *Multilayer Perceptron* (MLP). La idea de este modelo es incorporar una **capa oculta** de neuronas entre la capa de entrada y la de salida. La cantidad de neuronas que tendrá esta capa es determinada por el científico de datos. Ella aporta la capacidad que tiene la red de construir complejas funciones no lineales: mientras más neuronas en la capa oculta, mayor flexibilidad para ajustar una función compleja que minimice el error.

La flexibilidad de una red neuronal simple (con una sola capa oculta) es tal que puede aproximar cualquier función continua. Por esta razón, se dice que las redes son «aproximadores universales» capaces de aprender de los datos de forma perfecta (Minsky and Papert, 2017). El problema es que muchas veces aprenden el ruido intrínseco de los datos, llevándolas a sobreajustarse.

Otro aspecto importante que ha permitido el desarrollo de las redes neuronales es el aumento del poder de cómputo de los computadores. Los algoritmos han ido evolucionando con el tiempo, considerando cada vez más capas ocultas con los modelos de DL, lo que ha exigido una mayor exigencia de procesamiento.

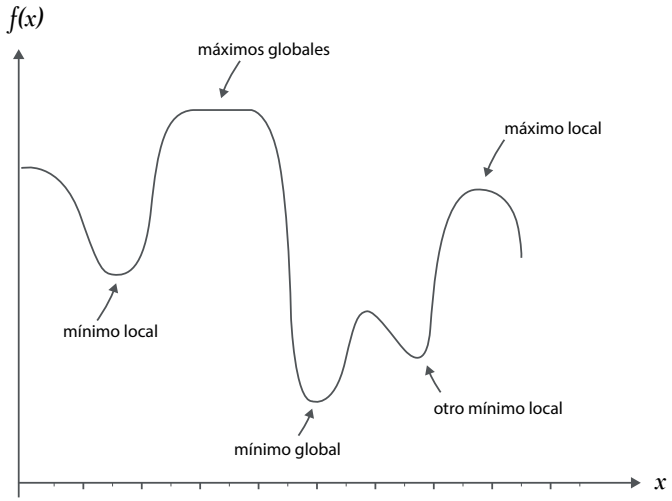
## 8.2 Algoritmo *backpropagation*

Este método permite encontrar una solución al problema de optimización que representa el entrenamiento de la red neuronal, es decir, permite calcular el error entre los valores obtenidos y los valores esperados por medio del cálculo de la función de pérdida, y minimizarlo.

La idea detrás de este algoritmo es propagar el error obtenido a través de la red, de manera de ajustar los pesos W de cada uno de los nodos para todas las capas. En este sentido, el algoritmo actualiza iterativamente los pesos «aprendiendo» del error cometido en la iteración anterior.

Una desventaja de esta estrategia es que suele encontrar **mínimos locales** como solución del problema. Dada la forma no lineal de la función asociada al modelo MLP, el problema de minimización que debe resolver *backpropagation* tiene una forma que se conoce como *no convexa,* con posibles mínimos locales (puntos de acumulación con forma de «U») pero con solo un único mínimo global, el cual es muy difícil de encontrar (Baesens, 2014). Esta forma se puede observar en la figura 8.2.

Figura 8.2: Visualización de la función de una red neuronal multicapa



El modelo *backpropagation* comienza con una ubicación al azar de las variables de la función y busca iterativamente realizar un salto en la dirección de descenso. Es como si tuviésemos una pelota saltarina que lanzamos en cualquier parte de la figura 8.2 y la dejamos descender, eventualmente se va a estancar en uno de los mínimos, que con un poco de suerte es el global, pero puede no serlo. El tema es que el algoritmo no tiene mecanismos para «salir del pozo» que representa un mínimo para buscar una solución mejor.

Una vez que definimos los pesos W iniciales de forma al azar (la ubicación inicial donde soltamos nuestra pelota saltarina), el algoritmo *backpropagation* repite iterativamente dos etapas: la fase *forward* («hacia adelante») y la fase *backward* («hacia atrás»).

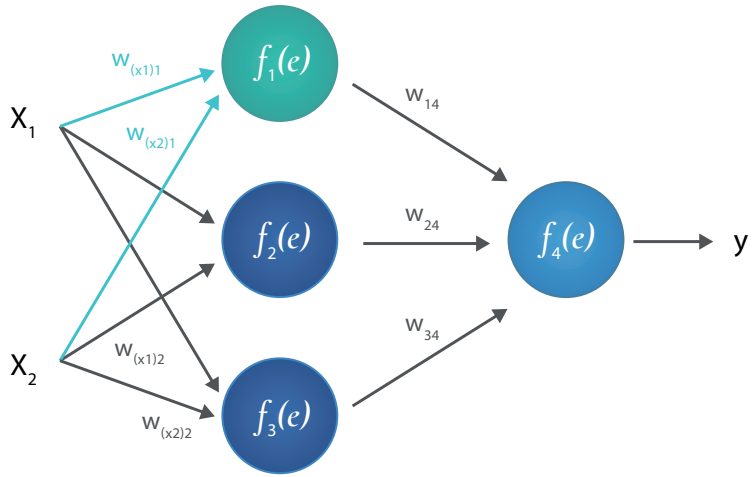
### *8.2.1 Fase forward*

En esta fase, los *inputs* recibidos en los nodos de entrada alimentan toda la red. Esto se realiza por medio del cálculo de una serie de operaciones algebraicas con el objetivo de evaluar la función completa, pasando por cada nodo y llegando a la evaluación de la función de pérdida.

Veamos el siguiente ejemplo[17] de una red de una capa oculta y dos variables predictivas ($x_1$ y $x_2$). La etapa *forward* alimenta con datos cada nodo de la capa oculta con tres neuronas, que corresponden a funciones *logit*. Por ejemplo, en la figura 8.3 se observa cómo se alimenta la primera neurona de la capa oculta.

[17] Ejemplo adaptado de M. Bernacki (2004), *Principles of training multi-layer neural network using backpropagation algorithm*. https://wiki.eecs.yorku.ca/course_archive/2013-14/F/4403/_media/backpropagation.pdf

Figura 8.3: Ejemplo ilustrativo de una red neuronal simple




El primer nodo queda de la siguiente forma:

Ecuación 8.2

$$y_1 = f_1(w_{(x1)1}x_1 + w_{(x2)1}x_2 + b)$$

Notar que la ecuación 8.2 muestra el equivalente a un *logit* si $f_1$ es una función sigmoidal. Además de los pesos, se incluye un intercepto $b$ (equivalente a $\beta_0$ en la ecuación 8.1). Una vez que se construyen las tres funciones de cada nodo ($y_1$, $y_2$ e $y_3$), la capa de salida combina estos tres *logits* en una función única de la siguiente forma:

Ecuación 8.3

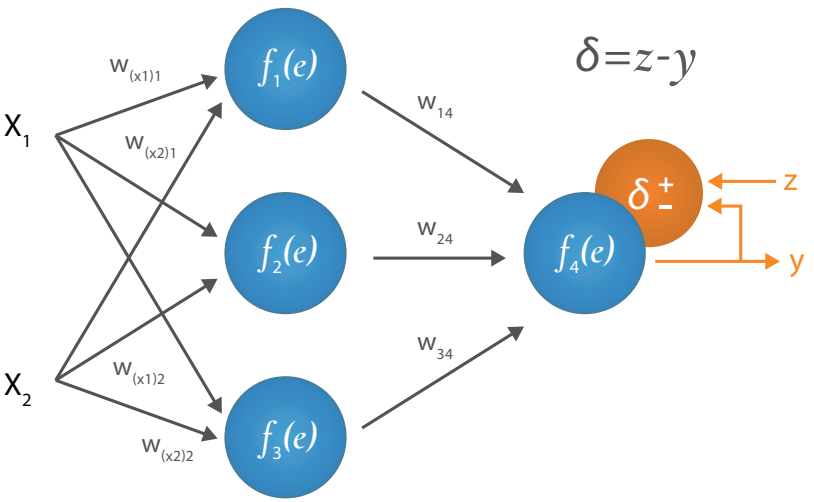
$$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3 + b)$$

La ecuación 8.3 muestra la forma funcional del modelo de red neuronal, que es «simplemente» un *logit* de funciones *logit*. Esto es importante, ya que mucha gente piensa que la modelación de caja negra es algo extremadamente «oscuro», casi como si el computador espontáneamente decidiera clasificar objetos y nadie supiese lo que pasa por detrás de las redes artificiales. Esto no es cierto, pero la analogía de las redes neuronales ciertamente levanta mitos que

se acercan a la ciencia ficción. Estos temas los discutiremos en el último capítulo del libro.

Volviendo al ejemplo, el último paso de la etapa *forward* es construir la función de pérdida, que corresponde a la diferencia entre nuestra predicción dada por $y_4$ y el valor real (que en la figura 8.4 llamamos $z$). Este error lo llamaremos $\delta$.

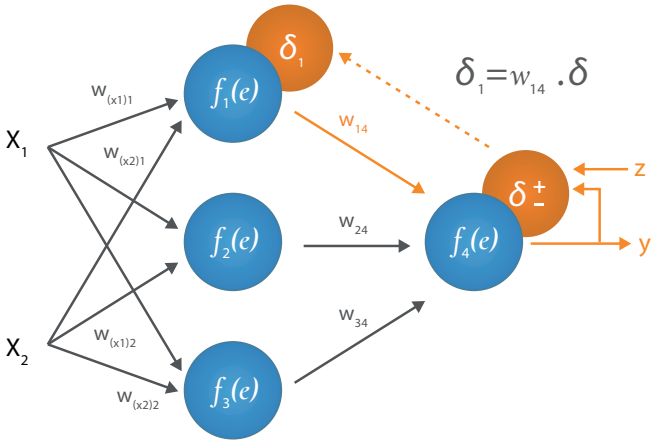
Figura 8.4: Cálculo del error y función de pérdida en una red neuronal artificial




## 8.2.2 *Fase backward*

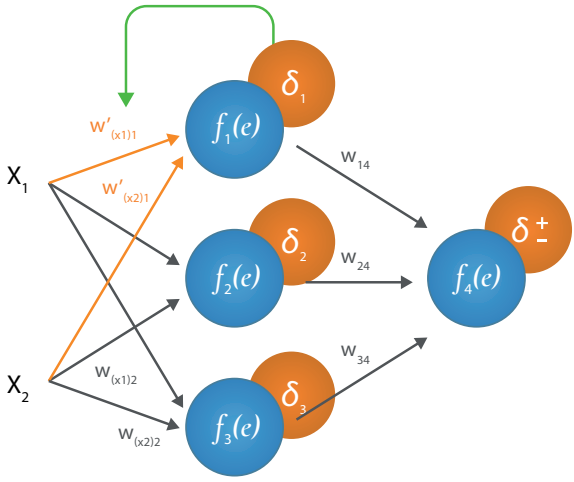
Esta fase se encarga del «aprendizaje» del modelo, el cual se realiza a través de la evaluación del gradiente de la función de pérdida en los diferentes pesos obtenidos en la fase *forward*. Una vez que tenemos nuestro error $\delta$, lo «propagamos hacia atrás» (por esto el algoritmo se llama así), es decir, lo repartimos a cada uno de los nodos de la capa oculta de acuerdo con los pesos del último *logit* combinador ($y_4$). Por ejemplo, llamamos $\delta_1$ a la fracción del error correspondiente al nodo 1 de la capa oculta, como se observa en la figura 8.5.

Figura 8.5: Propagación hacia atrás del error en una red neuronal




Una vez que todos los errores se obtienen, se procede a actualizar los pesos de la red. Como ya mencionamos, estos se generan al azar (posición inicial de nuestra pelota saltarina) y luego vamos dando saltos en la dirección de descenso. La dirección de descenso es precisamente el gradiente de la función de error. El gradiente es afín al concepto de derivada, pero desde un punto de vista multivariado (recordemos que tenemos muchos pesos que podemos ajustar para minimizar el error). Este ajuste se ilustra en la figura 8.6.

Figura 8.6: Actualización de pesos en una red neuronal



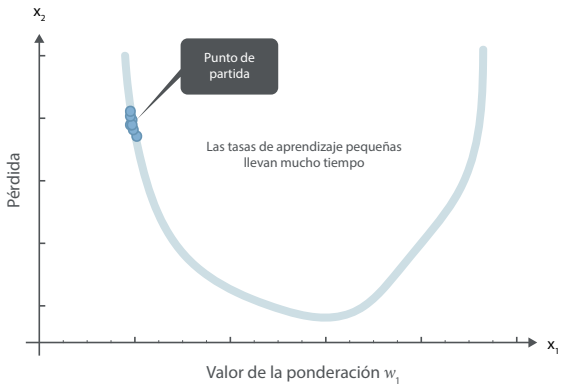
El nuevo peso $w'_{(x1)1}$ será entonces su valor anterior $w_{(x1)1}$ más un salto en la dirección de descenso (derivada de la función de pérdida), como se observa en la ecuación 8.4:

Ecuación 8.4

$$w'_{(x1)1} = w_{(x1)1}x_1 + \eta\delta_1 \frac{\partial f_4}{\partial w}$$

El término $\eta$ se conoce como **tasa de aprendizaje** y establece la magnitud del salto en la dirección de descenso (definida por la expresión que acompaña a $\eta$). Valores de $\eta$ muy pequeños pueden hacer muy lento el proceso de aprendizaje (figura 8.7[18]), mientras que valores muy grandes provocan saltos demasiado bruscos en los pesos y provocar que nuestra pelota «salte» el óptimo (figura 8.8). Estos fenómenos pueden afectar negativamente el desempeño del método. La figura 8.9 muestra una tasa de aprendizaje que logra balancear adecuadamente la posición anterior de los pesos y el aprendizaje (James *et al.*, 2017).

Figura 8.7: Tasa de aprendizaje muy pequeña




[18] Ejemplo adaptado de developers.google.com/machine-learning/

Figura 8.8: Tasa de aprendizaje muy grande

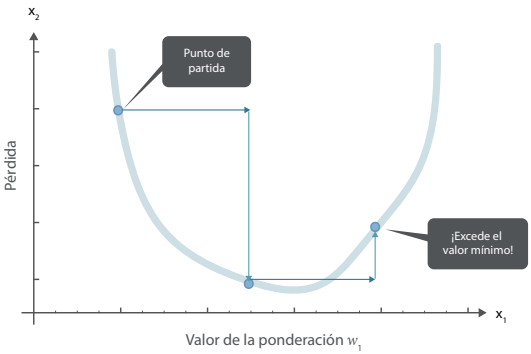
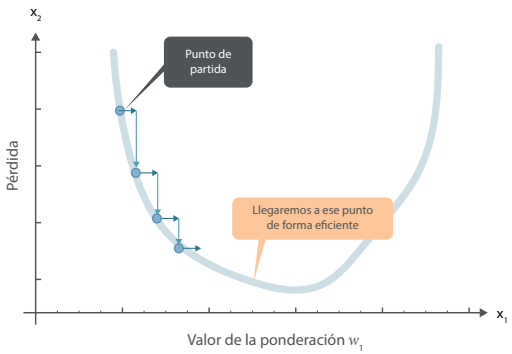



Figura 8.9: Tasa de aprendizaje adecuada




Muchas implementaciones consideran un parámetro extra llamado *momentum* ($\alpha$), el cual multiplica el primer término de la ecuación 8.4, quedando de la siguiente forma:

Ecuación 8.5

$$w'_{(x1)1} = \alpha w_{(x1)1} x_1 + \eta \delta_1 \frac{\partial f_1}{\partial w}$$

El *momentum* permite encontrar un balance entre la posición original y el salto en la dirección de descenso de forma más cómoda. Al disminuir la ponderación de la ubicación original del vector de pesos, el momentum permite que nuestra pelota «ruede» cada vez más rápido «cuesta abajo», adquiriendo momento (de aquí el nombre).

Notar que la inclusión de $\delta_1$ en las ecuaciones anteriores refleja este concepto de aprendizaje: cada peso es actualizado asumiendo su responsabilidad en el error del modelo, actualización que incorpora esta información con el fin de llegar a un menor error en la siguiente iteración. Este proceso iterativo se repite un número predeterminado de veces, que se conoce como número de épocas *(epochs)*.

Solo como acotación, el concepto detrás de este «aprendizaje» de las redes neuronales se basa en una técnica del cálculo diferencial llamada «regla de la cadena», que permite descomponer la derivada o gradiente de una función compuesta en la multiplicación de varias derivadas más simples, como se muestra en la ecuación 8.6:

Ecuación 8.6

$$\frac{d}{dt}f\big(g(t)\big) = \frac{df}{dg}\frac{dg}{dt} = f'\big(g(t)\big)g'(t)$$

### *8.2.3 Funciones de activación y pérdida*

Luego de entender la lógica del algoritmo de *backpropagation* y sus dos fases principales, es necesario aclarar un concepto fundamental dentro del entrenamiento, el cual es la función de activación: ¿para qué sirve? y ¿cómo elegirla? Notar que la sigmoidal no es la única opción.

La función de activación se aplica a los *inputs* de las neuronas de la red de dos posibles formas: previa a la suma de las entradas del nodo, para luego sumar los valores ($\Sigma f(WX)$, preactivación), o posterior a la suma de las multiplicaciones de los pesos con sus respectivas entradas ($f(\Sigma\, WX)$, post-activación).

Una tarea importante al momento de crear modelos de redes neuronales es la elección de las funciones de activación para cada una de las capas del modelo. Dentro del contexto de la programación de las redes, los avances y pruebas realizadas en la investigación de estos modelos han permitido identificar qué funciones de activación entregan un mejor rendimiento y robustez para determinados problemas.

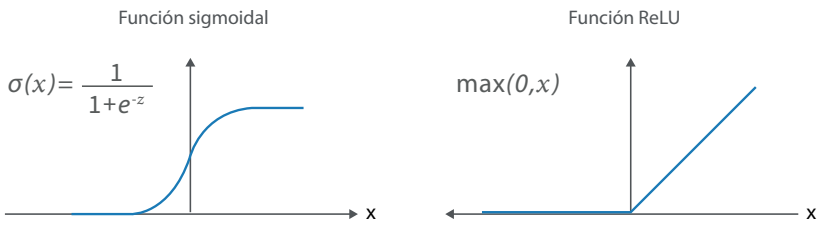
Dentro de la literatura, las funciones más utilizadas son las siguientes:

- **Sigmoidal:** corresponde a la función de activación más antigua y, antes de la aparición de ReLU, era de las más populares. Permite reducir el rango de valores de la entrada de datos entre 0 y 1, por lo cual convierte el *input* en una probabilidad. Era la opción natural si uno considera las redes neuronales como una extensión no lineal de la regresión logística.
- **Rectified Linear Unit (ReLU):** esta función transforma el *input* a valores no negativos simplemente tomando la parte positiva de la función. Es una función computacionalmente eficiente, ya que permite que la red converja muy rápido; además corresponde a una función no lineal, aunque parezca lineal, lo cual implica que tiene derivada y permite utilizar *backpropagation*. Existen variantes de ReLU como *Leaky ReLU* (ReLU con fugas), que corrigen sus desperfectos con los valores negativos. ReLU es de las funciones más populares a la hora de crear arquitecturas de DL.
- **Identidad:** esta función de activación no realiza ningún cambio en el rango de valores de entrada, ya que no realiza modificaciones en la entrada de datos. No aporta en la complejidad a la arquitectura del modelo, tan solo realiza la multiplicación $WX$. Es utilizada en situaciones en las que se desea que la salida del modelo sea del tipo regresión, es decir, para cuando las etiquetas son numéricas. Las tareas de regresión se profundizarán en la siguiente sección.

Otra ventaja de la función ReLU por sobre la sigmoidal (además de que es más eficiente) es que soluciona un problema específico que ocurre en DL, que tiene que ver con el cálculo del gradiente en arquitecturas profundas. La regla de la cadena (ecuación 8.6) implica muchas multiplicaciones cuando se consideran decenas o incluso centenas de capas ocultas. Esto genera inestabilidad en el cálculo del gradiente o dirección de descenso. En concreto, el gradiente se puede volver cero y el modelo no aprende, lo que se conoce como *vanishing*

*gradient*, o bien crecer indefinidamente *(exploding gradient)*. Dado que la derivada de la función ReLU es igual a 1, se vuelve mucho más estable para el cálculo del gradiente en problemas de DL. Por esto, es la opción preferida en modelos multicapa (Aggarwal, 2018). Las funciones sigmoidal y ReLU se presentan en la figura 8.10.

Figura 8.10: Funciones de activación sigmoidal y ReLU




La función de activación *softmax* corresponde a un caso especial de la función sigmoidal: extensión a tareas multiclase. En la ecuación 8.7 se puede observar la función *softmax*, la cual entrega la probabilidad $O_i$ asociada a cada clase $j \in \{1, \ldots, k\}$ para una observación $i$. Esta función posee una gran utilidad, ya que permite generar probabilidades de clasificación para cada una de las clases o etiquetas.

Ecuación 8.7

$$O_i = \frac{exp\ (v_i)}{\sum_{j=1}^{k} exp\ (v_j)}$$

Con respecto a las funciones de pérdida, una función común en clasificación se conoce como entropía cruzada o *cross-entropy loss*. Esta función toma la salida *softmax* o sigmoidal (probabilidad de ocurrencia de cada clase) y la relaciona con $z_j$ (la etiqueta de cada clase, 1 si $j$ es la clase correcta para la observación, 0 en caso contrario) de la siguiente manera:

Ecuación 8.8

$$L = \sum_{j=1}^{k} z_j \, log \, (O_j)$$

### *8.2.4 Método del gradiente estocástico*

La figura 8.2 nos muestra que el problema que resuelven las redes neuronales (minimización de una función de pérdida) es uno no lineal, con una alta probabilidad de «estancarse» en un mínimo local. El valor de la optimalidad global en redes neuronales no es muy alto, considerando que (1) el riesgo de sobreajuste de minimizar el error es alto y (2) el costo de «desestancarse» de un mínimo local y buscar diferentes soluciones es alto en términos computacionales. En este espíritu, ha habido una búsqueda constante de estrategias eficientes para el algoritmo *backpropagation,* que permitan al modelo aprender lo suficiente y llegar a una buena solución en un tiempo rápido.

La gracia de la estrategia de *gradiente de descenso estocástico* (*stochastic gradient descent* o SGD) es que permite hacer eficiente el cálculo de la dirección de descenso (ver ecuaciones 8.4 y 8.5). La idea es no calcular el gradiente completo, sino tomar solo un subconjunto de datos al azar. En este sentido, la dirección de descenso es una aproximación. De las diferentes variantes de SGD, la más popular hoy (año 2021) es *Adaptive Moment Estimation* o Adam. Sin embargo, es decisión del modelador elegir el optimizador para el problema pues existen otras opciones, tales como *Momentum-Based Learning, AdaGrad, RMSProp, AdaDelta,* entre otros.

Otra decisión del modelador asociada a SGD es el *tamaño de los batches*. Un *batch* es un lote de datos que se procesa en simultáneo y se utiliza para calcular las operaciones del modelo *backpropagation.* Este concepto está estrechamente ligado a requerimientos mínimos para los procesadores de la máquina (o múltiples máquinas) que se esté utilizando para entrenar el modelo. Tratando de llevarlo a términos sencillos, a un mayor tamaño de los *batches,* mayores serán los requisitos técnicos necesarios para computar dichas operaciones. Conceptos

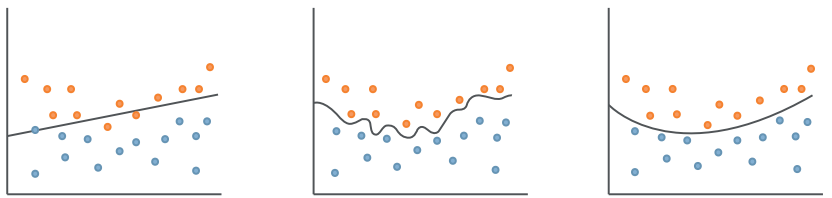
como GPU (unidad de procesamiento gráfico, especialmente útil para realizar operaciones matriciales y tensoriales), paralelismo (trabajo con múltiples procesadores en una tarea específica) y distribución de recursos de los computadores utilizados entran en juego a la hora de escoger el tamaño de los *batches*. Estos pueden generar cambios en los tiempos de entrenamiento, reduciéndolos o aumentándolos considerablemente.

### *8.2.4.1 Control del sobreajuste*

Como se indicó en los capítulos anteriores, el término *overfitting* hace referencia al problema de «memorización» que puede presentarse en las redes neuronales, en el cual el modelo se ajusta «demasiado bien» al conjunto de entrenamiento. En los modelos de redes neuronales el problema es grave debido a varias razones: (1) la arquitectura multicapa es muy flexible y no lineal («aproximador universal»), (2) el único objetivo de la red es tener el menor error posible (minimizar una función de pérdida) y (3) tiene una amplia gama de diferentes parámetros o «perillas» que podemos ajustar para obtener mejores resultados. Algunos de los mencionados hasta ahora son: tasa de aprendizaje, *momentum,* número de capas ocultas, número de neuronas en cada capa oculta, número de épocas y la elección de las funciones de activación y pérdida.

Nuestro objetivo es encontrar un nivel de no-linealidad balanceado, es decir, que el clasificador no sea demasiado simple (*underfitting* o subentrenado) o demasiado complejo (*overfitting* o sobreentrenado). Este *trade-off* se explica gráficamente en la figura 8.11.

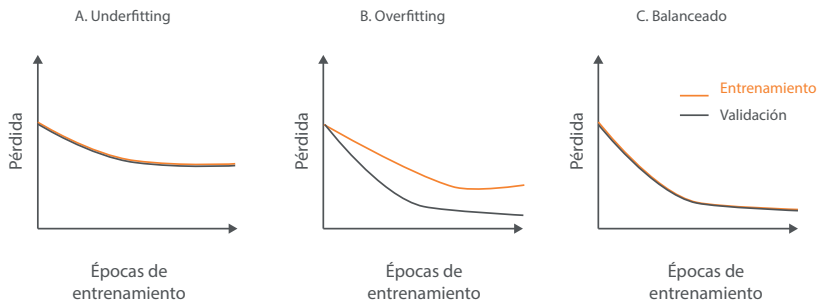
Figura 8.11: Representación gráfica del *overfitting* en un plano cartesiano de dos variables



Como se muestra en el primer gráfico de la figura 8.11, el patrón modelado es muy simple (lineal) y no logra capturar las complejidades del problema. Por el contrario, el segundo gráfico se ajusta demasiado a los datos de entrenamiento, mientras que el patrón es más sofisticado que la realidad misma. Esto provoca que el modelo no logre predecir de forma óptima nuevos ejemplos, pese a no cometer errores con los datos utilizados para entrenar. El tercer gráfico, en cambio, logra un balance correcto entre complejidad y ajuste a los datos (minimización del error).

Entendiendo los posibles estados de ajuste a los datos que posee un modelo, se vuelve necesario introducir una forma práctica de reconocer *overfitting* para los casos generales de múltiples dimensiones; de manera de identificarlo por medio de los gráficos de pérdida y *accuracy* de entrenamiento y validación. En la figura 8.12 se puede observar los tres casos que mencionamos anterior *(underfitting, overfitting, balanced)* para los gráficos de pérdida versus número de épocas:

Figura 8.12: Gráficos de pérdida versus número de épocas




En la figura 8.11 se puede observar que la pérdida durante el entrenamiento y la validación idealmente no debe distar excesivamente, sino que debería disminuir a la par. Otro signo de *overfitting* se puede observar en el segundo gráfico: si la pérdida es muy cercana a cero en una aplicación que sabemos que no es posible llegar a un desempeño perfecto (prácticamente cualquiera en el mundo de los negocios), es señal de que el modelo está aprendiendo del ruido intrínseco del problema (Kraus *et al.*, 2020).

Existen algunas soluciones para corregir el problema del sobreajuste en las redes neuronales, las cuales se detallan a continuación:

- ***Early stopping*:** una solución simple es detener tempranamente el entrenamiento, es decir, aplicar *backpropagation* un número reducido de iteraciones o épocas, monitoreando la función de pérdida como en la figura 8.12. En el momento en que la pérdida no se reduce significativamente de iteración en iteración, paramos el proceso. Esto puede ocurrir incluso en un número tan pequeño como 10 iteraciones.
- **Regularización y *weight decay*:** el objetivo de la regularización es reducir el tamaño de los pesos de la red como una forma de controlar el aprendizaje. Esto transforma el problema de optimización en uno con objetivo doble: por un lado, minimizar el error y, por otro, mantener los pesos pequeños, lo que está demostrado que ayuda a reducir el sobreajuste. Los dos tipos de regularización más comunes son la L1 y L2, que corresponden a la minimización de la suma de los pesos en valor absoluto y al cuadrado, respectivamente. En este sentido, L1 y L2 están vinculadas con las medidas de distancias Manhattan y Euclideana, presentadas en capítulos anteriores. El parámetro que acompaña la regularización se conoce como decaimiento de pesos o *weight decay*, y controla el balance entre ajuste y complejidad. Este parámetro lo debe definir el modelador. Por ejemplo, una función de pérdida con *cross-entropy* y regularización L2 se vería de la siguiente manera ($wd$ representa el decaimiento de pesos y se asumen $L$ pesos $w$ en la red):

Ecuación 8.9

$$L = \sum_{j=1}^{k} z_j \, log\,(O_j) + wd \sum_{l=1}^{L} w_l^2$$

- ***Dropout:*** técnica de *ensemble* que elimina selectivamente neuronas para crear diferentes redes simples. Finalmente se combinan en una. Esto es diferente a la regularización L1 que es capaz de eliminar pesos de la red. Con *dropout* descartamos neuronas completas.

En resumen, las redes neuronales tienen importantes ventajas y desventajas. La principal ventaja es su flexibilidad, que permite modelar de mejor manera datos complejos, llegando a desempeños predictivos potencialmente mejores que otros métodos. Pero la flexibilidad es un arma de doble filo, y es también su principal debilidad:

- La no-linealidad del método obliga a sacrificar la interpretación de los coeficientes con la que gozamos en regresión logística.
- La optimización es compleja y no-lineal, lo que puede ralentizar el método. Además, no se asegura que se llegue a una solución de mínimo global. Afortunadamente, el modelo funciona bastante bien cuando aprende un cierto número de iteraciones, sin la necesidad de preocuparse demasiado por este tema, que es, más bien, un debate matemático.
- Debido al punto anterior, el modelo es poco robusto en el sentido de que diferentes condiciones iniciales suelen llevar a diferentes soluciones. Recordemos que los pesos se inicializan de forma aleatoria y, por ende, las soluciones a las cuales las redes pueden llegar, si corremos el modelo varias veces, pueden ser muy diferentes.
- La cantidad de parámetros o «perillas» es grande, por lo que evaluar diferentes combinaciones mediante una lógica de «prueba y error» puede ser computacionalmente muy costoso. Una técnica común para la elección de parámetros se conoce como búsqueda en cuadrícula (*grid search*), en donde se define un conjunto de candidatos para cada parámetro y se evalúan las diferentes combinaciones.

Como recomendación para la implementación exitosa, confiar en las configuraciones por defecto de las diferentes implementaciones de las redes neuronales hasta ganar la experiencia necesaria para manipular los diferentes parámetros es importante. Las implementaciones más conocidas en softwares como R, Python o SAS recogen «buenas prácticas» y experiencias de la comunidad, siendo la mejor apuesta inicial para encontrar buenos resultados sin la necesidad de costosas evaluaciones de parámetros. Como segunda recomendación, siempre escalar los datos para que estén en una escala común y filtrar atributos irrelevantes para reducir el ruido en los datos, lo que ayuda a evitar sobreajuste.

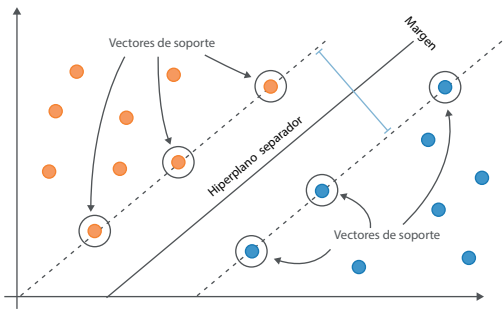
### *8.2.5 Relación con Support Vector Machines*

Al momento de introducir los modelos de redes neuronales, se vuelve necesario realizar un paréntesis para mencionar otra clase de modelos del mundo de *machine learning,* que tienen una estrecha relación con las redes neuronales, ya que también se relacionan con problemas de clasificación y regresión. Estas son las Máquinas de Soporte Vectoriales o *Support Vector Machines* (SVM).

Los modelos de SVM corresponden a modelos de aprendizaje supervisado y, al igual que las redes neuronales, son capaces de construir una función no lineal que separe los datos. SVM se puede considerar como un caso particular de red neuronal, en donde se considera regularización L2 para encontrar un adecuado compromiso entre el ajuste del modelo y reducir la complejidad de la función. La interpretación geométrica de SVM es que maximiza el margen de separación entre las dos clases, lo que conlleva a un modelo predictivo más robusto y menos complejo.

El principio de parsimonia detrás de SVM sugiere que la regularización L2 hace que la función no lineal se «soporte» en menos puntos. De esta forma, la función no tiene tantos recovecos y es menos propensa a sobreajustarse (recordar figura 8.11). Dado que cada punto es un vector (en nuestros ejemplos, un cliente), la función se «soporta» en los llamados vectores soportantes *(support vectors)*, que le dan el nombre al método. El margen de separación máximo define los vectores soportantes y, una vez que se tiene este margen máximo, la función no lineal pasa por el medio del margen. Este fenómeno se ilustra en la figura 8.13.

Figura 8.13: Ejemplo de SVM en un plano cartesiano de dos variables



La no-linealidad de SVM viene dada por una *función de kernel*. Geométricamente hablando, esta función proyecta los datos a un espacio de mayor dimensión, en donde se maximiza el margen y se estima un modelo lineal. Luego se lleva este modelo al espacio original, resultando en una superficie no lineal.

SVM fue diseñado a comienzos de los noventa para hacerse cargo de las principales desventajas de las redes neuronales de los ochenta manteniendo la principal ventaja de la flexibilidad. El modelo SVM tiene las siguientes características con relación a las redes neuronales:

Es un modelo no lineal, flexible y con potencial de llegar a muy buenos resultados predictivos.

Su estructura biobjetivo de maximización del margen y ajuste de datos de forma simultánea reduce el riesgo de sobreajuste con relación a las redes. SVM plantea un modelo de optimización convexo, es decir, que solo existe un mínimo global. Esto asegura que la solución del modelo sea única y la mejor posible, y evita tener que definir condiciones iniciales de pesos que puedan llevar a resultados diferentes.

Tiene menos parámetros que las redes, lo que facilita su calibración. Mientras que en redes contamos más de seis parámetros, con SVM tenemos solo dos: un parámetro de balance que se representa con la letra C (equivalente a *weight decay*) y un parámetro que define la forma del kernel que se representa con las letras griegas $\gamma$ o $\sigma$, dependiendo de la implementación.

El método SVM cobró gran popularidad en los noventa y comienzos del nuevo siglo. Tenía la batalla prácticamente ganada en comparación con las redes neuronales. Sin embargo, las redes profundas mostraron una efectividad superior en tareas como análisis de imágenes, texto o señales. A continuación, discutiremos las arquitecturas más importantes de DL, cerrando este recorrido histórico

de métodos de clasificación con los avances más importantes de la última década.

## 8.3 Actualidad en los modelos de *deep learning*

Gracias a los avances de la tecnología, en conjunto con el desarrollo de modelos y técnicas para captar patrones más complejos, se ha logrado realizar avances impensados pocas décadas atrás.

Los avances recientes de DL han generado gran interés en universidades, empresas e institutos de investigación de todo el mundo. Algunos de los modelos más importantes son: Redes Neuronales Convolucionales (CNN, de las siglas en inglés), Redes Neuronales Recurrentes (RNN, de las siglas en inglés), modelos basados en atención y *transformers,* modelos generativos y *Generative Pre-trained Transformer* (GPT-3).

A continuación, se describirá en qué consisten, su utilidad y rasgos más importantes:

### *8.3.1 Convolutional neural network (CNN)*

Las redes neuronales convolucionales corresponden a una clase de redes profundas (es decir, que tienen muchas capas), comúnmente aplicadas para realizar diversos tipos de análisis de imágenes. Las CNN se utilizan principalmente en el reconocimiento de imágenes y video, sistemas de recomendación, clasificación de imágenes, análisis de imágenes médicas, procesamiento de lenguaje natural y series de tiempo financieras (LeCun, 2015).

Un rasgo característico de este tipo de redes es que no son *fully-connected,* es decir, que todas las neuronas en una capa están conectadas a todas las neuronas de la capa siguiente. La racionalidad es tener diferentes capas que se enfoquen en diferentes partes de la imagen. Para esto no es necesario que todas las neuronas se conecten entre sí. Esto, a diferencia de las redes neuronales de las que hablamos anteriormente, que eran todas *fully-connected.*

Otro concepto importante es el de una **capa de convolución**, el cual busca rescatar «cuadros» o porciones de la imagen original, para de esta forma identificar objetos en las imágenes de forma más precisa. Esto se realiza a través de un *kernel*, que es un filtro que rescata características o *features* de una imagen.

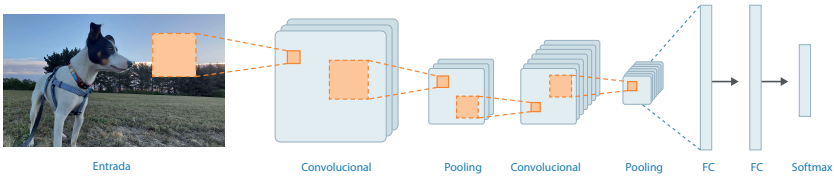
Antes de las redes convolucionales, el proceso de extracción de *features* solía hacerse de forma independiente al aprendizaje: primero se aplicaban métodos como Análisis de Componentes Principales para extraer características de las imágenes, luego se alimentaba una red neuronal u otro modelo con estos *features*. La «magia» de las redes CNN es que permite «aprender» de forma supervisada la mejor forma de extraer *features*, es decir, la extracción de características y la clasificación se realiza en el mismo proceso sin perder información.

Después de una capa de convolución suele venir una capa de *pooling*, la cual resume las convoluciones en imágenes más pequeñas preservando las características más importantes de los «cuadros» obtenidos en la capa anterior. Este proceso se conoce como *sub-sampling* o muestreo. El más común suele ser *max-pooling*. Por ejemplo, *max-pooling* selecciona un cuadrado de 2x2 de la capa de convolución y toma el máximo de los cuatro posibles valores.

Una red convolucional tradicional suele contar con varias de convolución y *pooling* de manera alternada, para terminar con una capa *fully-connected* que rescata toda la información disponible y una capa de salida (comúnmente una *softmax*, ya que detección de imágenes suele ser una tarea multiclase).

Un ejemplo de una red convolucional se presenta en la figura 8.14. Es importante notar que existe una amplia variedad de redes CNN, ya que su éxito empírico ha generado un sinnúmero de variantes.

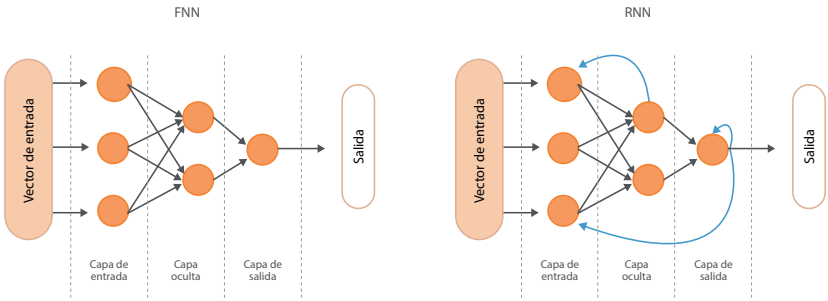
Figura 8.14: Ejemplo de una red convolucional tradicional




## *8.3.2 Recurrent Neural Networks (RNN)*

Esta clase de redes viene a cambiar el paradigma clásico de funcionamiento de las redes neuronales, ya que considera como entrada otras capas de la misma red de manera recurrente. Esto se ilustra en la figura 8.15 con las flechas que apuntan hacia atrás.

Figura 8.15: Ejemplo de una red recurrente en contraste con una red tradicional




Este tipo de redes permite modelar problemas complejos que requieren modelar la secuencia de las observaciones (imágenes, textos y audio comúnmente) para comprender su significado correcto. Para ello considera la temporalidad o instante de tiempo en que ingresa cada componente del vector. Un ejemplo puede ser un texto compuesto por palabras que siguen una cierta secuencia dentro de la oración. El enfoque tradicional en aprendizaje de máquinas era sacrificar el orden de las palabras para interpretar texto, pero con las redes RNN fue posible rescatar esta información y «aprender» directamente la mejor representación matemática de un texto o *embedding*.

Una red RNN agrega memoria a cada neurona para usarla como una entrada adicional en la próxima ejecución. Sin embargo, este enfoque tiene algunos problemas. Primero, si las neuronas tienen que recordar todos sus resultados pasados, la cantidad de conexiones en la red será extremadamente grande. Además, cuando una red neuronal no puede olvidar, no puede aprender cosas nuevas tampoco.

La primera solución que se propuso fue limitar la memoria de una red neuronal. Por ejemplo, que memorice no más de cinco valores recientes. Sin embargo, esto no funcionó como se esperaba ya que existe un valor importante en la memoria de largo plazo de las RNN.

La siguiente solución fue mucho mejor: diseñar celdas especiales, similares a la memoria de la computadora. La celda, que viene a reemplazar un nodo tradicional, puede registrar un número, leerlo o restablecerlo. Se les llamó células de memoria a largo y corto plazo (*long short-term memory* o LSTM). En otras palabras, la red está capacitada no solo para aprender ponderaciones sino también para establecer recordatorios, manejando una memoria de corto plazo y una de largo plazo. De esta forma se pudo mantener la ventaja original de las RNN, pero de forma mucho más eficiente en términos computacionales. En la misma dirección nacen en el año 2014 las *gated recurrent units* (GRUs), que son variaciones de LSTM, pero con menos parámetros, lo que conlleva a un entrenamiento más rápido y un menor riesgo de sobreajuste.

El siguiente paso en la evolución de las redes recurrentes fue el concepto de *transformers*. Estos corresponden a una clase de redes neuronales que se denominan arquitecturas de «secuencia a secuencia» (o Seq2Seq), las cuales son utilizadas en tareas de NLP. Estos modelos transforman una secuencia de elementos, como la secuencia de palabras en una oración, en otra secuencia que puede ser comprendida por el computador. De momento se ven bastante similares en comparación al funcionamiento de las RNN, pero, a diferencia de estas últimas, los *transformers* poseen un mecanismo de atención encargado de procesar el *input* de texto y determinar qué partes de la secuencia son importantes.

Los t*ransformers* fueron introducidos por Google el año 2017 y diseñados para trabajar con datos secuenciales enfocados en generar una salida diferente a los datos de entrada. Algunos ejemplos son traducir un texto de un idioma a otro, completar una oración inconclusa, clasificar los sentimientos de un determinado texto (*tweets* u opiniones de un restorán) o también la clasificación de texto en múltiples categorías (*reviews* de hoteles, por ejemplo).

En términos sencillos, el mecanismo de atención se encarga de mapear las palabras importantes y relevantes de un *input* de texto, y asignar un mayor peso a estas palabras. Esta cualidad permite mejorar el *Accuracy* de la red. El mecanismo ha sido sometido a múltiples actualizaciones y variaciones para adaptarse a diferentes tareas.

El modelo *Bidirectional Encoder Representations From Transformers* o BERT, corresponde a un modelo de atención y perteneciente a los *transformers*. BERT cambió el paradigma de los modelos de NLP anteriormente utilizados para el procesamiento de secuencias, ya que dejó de «leer» los textos de izquierda a derecha, como mencionamos primeramente en las RNN. Los *inputs* se consideran de manera bidireccional a través de la novedosa técnica denominada *masked language modeling* (MLM). Como se indicó en la aplicación 4, la idea es enmascarar alrededor del 15% de las palabras en una secuencia u oración, luego busca predecir estas palabras faltantes para «entender el contexto» (Devlin *et al.*, 2018).

BERT aplica también el concepto *next sentence prediction* (NSP), que permite extraer contexto entre oraciones. El modelo recibe pares de oraciones como entrada y busca predecir si la segunda oración es subsecuente a la primera en el documento original. El 50% de las entradas son pares de oraciones consecutivas del documento original, mientras que el otro 50% son oraciones al azar. Bajo el supuesto de que dos oraciones al azar están desconectadas, si el modelo logra aprender a identificar cuando las oraciones son consecutivas, entonces es capaz de tener una comprensión mayor de párrafos de varias oraciones.

Puede parecer demasiado ambicioso ser capaz de identificar palabras faltantes en una oración o identificar secuencias de oraciones.

Para ser honestos, lo es: es muy difícil calibrar un modelo así con textos cortos. La gracia es que BERT utiliza el concepto de *transfer learning,* el cual corresponde a la importación de un modelo previamente entrenado con una cantidad masiva de datos de texto. Por ejemplo, BERT cuenta con un modelo preentrenado a partir de libros (800 millones de palabras en BooksCorpus) y Wikipedia (2500 millones de palabras) (Annamoradnejad, 2020). Este modelo preentrenado se adapta a los textos de la aplicación en la cual se está trabajando en la etapa de *fine-tuning*.

### *8.3.3 Modelos generativos y GPT-3*

Los modelos generativos corresponden a un tipo especial de aprendizaje diferente a la clasificación que se enfocan en aprender en base a diferentes tipos de datos y generar nuevos ejemplos con ligeras variaciones, a partir de los datos de entrenamiento. Al igual que los modelos anteriores, requiere de conjuntos de datos muy grandes para toda clase de datos (data estructurada, imágenes, texto, voz). Por ejemplo, estos modelos se utilizan para crear nueva música o incluso obras de arte en base a datos de audio o imágenes.

Dentro de las arquitecturas comunes de DL para modelos generativos encontramos las *generative adversarial networks* (GANs), las cuales buscan entrenar dos modelos en paralelo. Ambas arquitecturas se alimentan mutuamente: uno, por un lado, es un modelo generativo G que captura la distribución de los datos y genera muestras a partir de un modelo estadístico, mientras que existe otro modelo discriminador D que se comporta como clasificador.

El *generative pre-trained transformer 3* (GPT-3) es la tercera versión de una serie de modelos generativos lanzados por OpenAI, un laboratorio de investigación en IA que tiene a Elon Musk como propietario. GPT-3 es un modelo de lenguaje para completar oraciones (como la función de autocompletar de Google, aunque mucho más avanzado) que ha logrado avances impresionantes, tales como generar contenidos a partir de órdenes sencillas en inglés. Por ejemplo, ha programado aplicaciones en lenguajes de programación

como React o HTML/CSS sin la necesidad de realizar actualizaciones del gradiente y los parámetros del modelo. Otra aplicación muy comentada es la de crear artículos y notas para las que es muy difícil de distinguir si fueron escritas por humanos, o no. Al igual que BERT, se basa en un modelo del tipo *transformer* preentrenado con millones de palabras, con la diferencia de que busca generar contenido en vez de clasificar texto.

## Referencias


Aggarwal, C. C. (2018). *Neural Networks and Deep Learning. A Textbook*. New York: Springer Verlag.

Annamoradnejad, I. (2020). ColBERT: *Using BERT Sentence Embedding for Humor Detection*. arXiv:2004.12765.

Baesens, B. (2014). *Analytics in a Big Data World*. New Jersey: John Wiley and Sons.

Devlin, J., Chang, M. -W., Lee, K. & Toutanova, K. (2018). Bert: *Pretraining of deep bidirectional transformers for language understanding*. ArXiv.cs.CL 1810.04805.

Han, J., Kamber, M. & Pei, J. (2011). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Kraus, M., Feuerriegel, S. & Oztekin, A. (2020). «Deep learning in business analytics and operations research: Models, applications and managerial implications». *European Journal of Operational Research,* 281(3), 628-641.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). «Deep learning». *Nature*, 521(7553), 436-444.

Minsky, M. & Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry.* Cambridge, MA: MIT press.

Rosenblatt, F. (1958). «The perceptron: A probabilistic model for information storage and organization in the brain». *Psychological Review,* 65(6), 386.

# Laboratorio 3: Clasificación

En este laboratorio haremos una comparación de los diferentes modelos de clasificación vistos en el libro para la base de riesgo de crédito. El objetivo es encontrar el mejor modelo, para aplicarlo en modo predictivo en el siguiente taller.

Primero importamos las librerías a utilizar (aquí se muestran solo algunas librerías; el detalle completo está en el sitio web del libro):

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier, kneighbors_graph
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
```

Como se observa, los clasificadores provienen de la librería *sklearn,* que es la base para correr modelos de aprendizaje de máquinas en Python.

El siguiente paso es montar Google Drive para enlazar nuestra base de datos. De esta forma podremos cargar después nuestra base de datos creditSIM.xlsx (que debemos tener en una carpeta «Colab Notebooks» en nuestro Drive claramente identificada).

```
from google.colab import drive drive.mount('/content/gdrive')
```

Cargamos la base preprocesada del Laboratorio 1:

```
ruta_data_set = 'gdrive/My Drive/creditSIM_preprocess.xlsx'
data_set = pd.read_excel(ruta_data_set)
```

## 1. Separación de datos

Definimos la variable dependiente como «Y» y las variables independientes como «X»:

```
y = data_set['VarObj_S']
del data_set['VarObj_S'] X = data_set
```

Separamos el total de datos en datos de entrenamiento (70%) y de testeo (30%):

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Como es común en riesgo de crédito, esta base presenta un desbalance de clases. Corregiremos el problema en el proceso de validación de modelos. Primero vemos cuántas personas de nuestra base de datos caen o no en *default:*

```
print(y_train.value_counts())
```

```
0       1664
1       436
```

Es importante notar que tenemos un desbalance, ya que la tasa de clientes malos pagadores es cercana al 20%. Para corregir este desbalance utilizaremos SMOTE, un método muy popular de resampleo de datos que genera ejemplos sintéticos de la clase minoritaria.

Nota: esta base adaptada solo será utilizada en regresión logística, mientras que en los restantes métodos utilizaremos la base original y le diremos al código que aplique SMOTE dentro de la validación.

```
sm = SMOTE()
X_train_n, y_train_n = sm.fit_resample(X_train, y_train)
```

Vemos la cantidad de personas que hay en cada clase de la variable objetivo para comprobar que están parejas:

```
unique, counts = np.unique(y_train_n, return_counts=True)
dict(zip(unique, counts))
```

```
{0: 1664, 1: 1664}
```

Por último, transformamos en un *dataframe* nuestros nuevos X de entrenamiento y testeo:

```
X_train_n = pd.DataFrame(X_train_n, columns = X.columns)
X_test = pd.DataFrame(X_test, columns = X.columns)
```

## 2. Entrenamiento y validación

En esta sección construiremos los modelos con los datos de entrenamiento. Se realiza validación cruzada con el objetivo de encontrar los mejores parámetros para cada modelo.

### *2.1 Regresión logística*

Definimos el modelo de regresión logística con nuestros X e Y de entrenamiento resampleados con SMOTE (estamos utilizando la librería *statsmodels*):

```
logit_model=srm.Logit(y_train_n, srm.add_constant(X_train_n))
result=logit_model.fit()
print(result.summary2())
```

```
                Results: Logit
============================================================
Model:          Logit                Pseudo R-squared: 0.509
No. Observations: 3328               Log-Likelihood:   -1132.5
Df Model:       7                    LL-Null:          -2306.8
Df Residuals:   3320                 LLR p-value:      0.0000
Converged:      1.0000               Scale:            1.0000
No. Iterations: 8.0000
------------------------------------------------------------
               Coef.   Std.Err     z     P>|z|   [0.025  0.975]
------------------------------------------------------------
const          7.1962   1.3009  5.5318  0.0000  4.6465  9.7459
AgnosDirec    -0.1816   0.0169 -10.7737  0.0000 -0.2146 -0.1486
AgnosEmpleo   -0.3088   0.0210 -14.6777  0.0000 -0.3500 -0.2675
Edad          -0.0353   0.0069  -5.1374  0.0000 -0.0487 -0.0218
DeudaIng       4.6123   1.3871   3.3252  0.0009  1.8937  7.3309
```

```
Ln_Ingreso2     -2.2836     0.2065    -11.0577     0.0000     -2.6884     -1.8789
Ln_DeudaExt      0.4963     0.1254      3.9570     0.0001      0.2505      0.7422
Ln_DeudaInt      2.2798     0.1117     20.4081     0.0000      2.0609      2.4988
==================================================================================
```

Notemos que todas las variables tienen un p-valor menor a 0.05 (ver antepenúltima columna), por lo que son todas relevantes. En otras palabras, no es necesario eliminar ninguna.

Alternativamente, podemos construir la regresión logística con la librería *sklearn*, la cual utilizaremos para todos los modelos. Esta última puede ser más robusta en temas de convergencia, pero no logra entregar los p-valores de la regresión, solo los coeficientes.

```
regresion = LogisticRegression(random_state=0)
regresion.fit(X_train_n, y_train_n)
pd.DataFrame(zip(X_train_n.columns,np.transpose(regresion.coef_.tolist()
[0])),column=['features','coef'])
```

Out[12]:

| | features | coef |
|---|---|---|
| 0 | AgnosDirec | -0.172509 |
| 1 | AgnosEmpleo | -0.301058 |
| 2 | Edad | -0.033524 |
| 3 | DeudaIng | 1.864653 |
| 4 | Ln_Ingreso2 | -2.470237 |
| 5 | Ln_DeudaExt | 0.661312 |
| 6 | Ln_DeudaInt | 2.322569 |

Si comparamos los coeficientes del modelo utilizando *statsmodels* y *sklearn*, notamos que son bastante parecidos. En principio podemos utilizar cualquier versión.

## *2.2 Árbol de decisión*

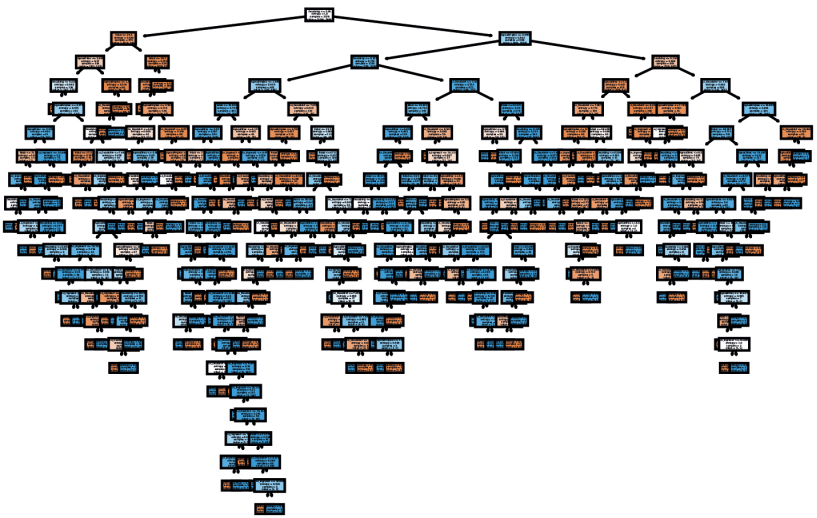
Primero, definimos el método de clasificación (árbol de decisión basado en entropía):

```
arbol = DecisionTreeClassifier(criterion = 'entropy')
```

Si queremos el árbol sin ningún parámetro de poda, entrenamos el modelo utilizando la configuración por defecto con la matriz X y las etiquetas Y de entrenamiento (resampleadas). Entonces graficamos el árbol obtenido:

```
arbol.fit(X_train_n, y_train_n)
tree.plot_tree(arbol, feature_names = data_set.columns, filled = True)
plt.show()
```

Figura 1: Árbol sin parámetros de poda



Es posible distinguir que el árbol anterior está claramente sobreajustado debido a su profundidad, lo que lo hace no interpretable y, probablemente, subóptimo en términos de desempeño predictivo. Para evitar este problema, estableceremos diferentes criterios de poda para obtener un árbol más pequeño y fácil de interpretar. Evaluaremos el modelo con los diferentes parámetros de poda utilizando validación cruzada. Con este procedimiento reducimos el riesgo de sobreajuste.

- *max_leaf_nodes:* número máximo de nodos «hoja» en el modelo (nodos terminales).
- *max_depth:* profundidad máxima del árbol.

```
max_leaf_nodes = [5,10,20,25]
max_depth = [3,4]
parametros = dict(tree_max_leaf_nodes = max_leaf_nodes, tree_max_
depth=max_depth)
```

Utilizamos la opción *pipeline* para construir los diferentes árboles utilizando SMOTE para corregir el desbalance de clases. La opción *GridSearchCV* es la que realiza validación cruzada de 10 particiones (cv=10) para los diferentes conjuntos de parámetros. Esto no se hace para la regresión logística, ya que esta última no tiene parámetros calibrables.

```
model_tree = Pipeline(steps=[('sampling', SMOTE()),
('tree', DecisionTreeClassifier(criterion = 'entropy'))])
tree_cv=GridSearchCV(model_tree,parametros,cv=10)
 tree_cv.fit(X_train, y_train)
```

Vemos cuáles son los mejores parámetros según la grilla:

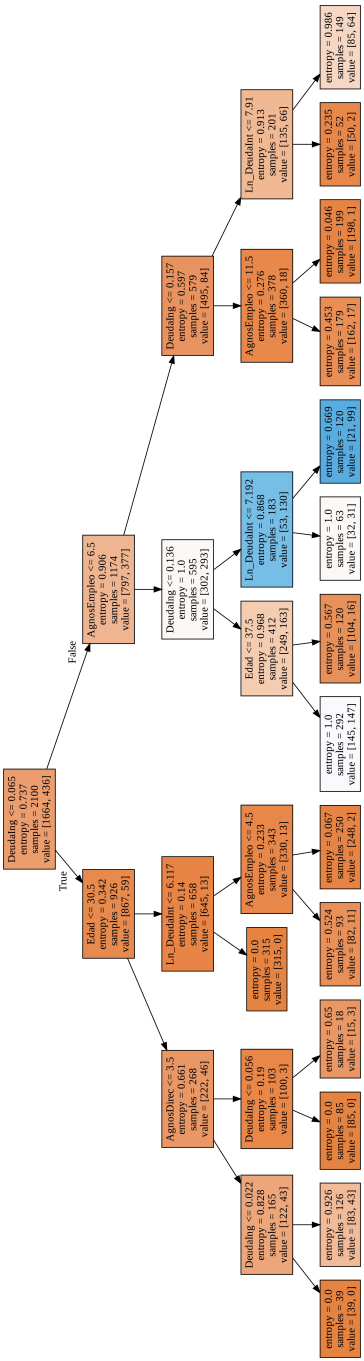
```
print('Best params:')
print(tree_cv.best_params_)
```

```
Best params:
{'tree_max_depth': 4, 'tree_max_leaf_nodes': 25}
```

Graficamos nuestro nuevo árbol:

```
arbol = DecisionTreeClassifier(criterion = 'entropy',max_depth= 4, max_leaf_no-
des= 20
arbol.fit(X_train, y_train)
#con estas funciones se grafica más grande y es más fácil de leer
dot_data = StringIO()
export_graphviz(arbol, out_file=dot_data, feature_names=data_set.
columns,filled= True)
(graph, ) = graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Figura 2: Árbol post SMOTE



Este árbol sí es interpretable. En él podemos observar que el ratio deuda sobre ingreso es la variable más relevante, ya que se utiliza para la primera ramificación. Si el endeudamiento es bajo, el árbol sugiere ramificar por edad, mientras que la variable años en el mismo empleo es la elegida para ramificar si el endeudamiento es alto. Los nodos y hojas en color naranjo tienen una proporción alta de clientes buenos, mientras que los que están en celeste tienen una mayor proporción de clientes malos. En blanco se encuentran los nodos con proporción 50-50 (entropía cercana a 1).

Finalmente, guardamos el mejor modelo de la grilla para luego obtener sus resultados de predicción:

```
arbol = tree_cv.best_estimator_
```

Los modelos estudiados a continuación requieren que los datos se encuentren escalados entre 0 y 1, por lo que normalizaremos la base de datos:

```
scaler = MinMaxScaler()
scaler_data = scaler.fit(data_set)
data_set_scaled = pd.DataFrame(scaler_data.transform(data_set), index=data_
set.index, columns=data_set.columns)
```

Dividimos nuevamente los datos:

```
X_train_scaled,X_test_scaled,y_train_scaled,y_test_scaled= train_test_split(data_
set_scaled, y, test_size=0.3, random_state = 0)
```

## *2.3 K-vecinos más cercanos*

Definimos el método de clasificación a utilizar:

```
Knn = KNeighborsClassifier()
```

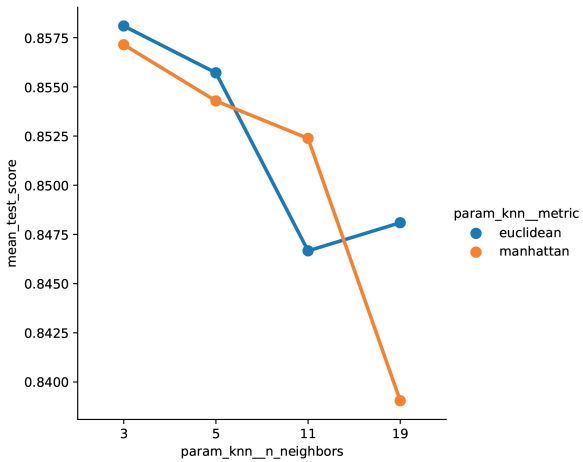
Definimos los parámetros para la grilla:

- *n_neighbors:* número de vecinos.
- *metric:* métrica de distancia que se utiliza para el árbol (Euclideana o Manhattan).

```
n_neighbors= [3,5,11,19]
metric = ['euclidean','manhattan']
parametros= dict(knn_n_neighbors= n_neighbors,
knn_metric= metric)
 model_knn = Pipeline(steps=[('sampling', SMOTE()), ('knn', Knn)])
knn_cv=GridSearchCV(model_knn,parametros,cv=10)
knn_cv.fit(X_train_scaled, y_train_scaled)
#Vemos los mejores parámetros según la grilla
print('Best params for k-neighborns:')
print(knn_cv.best_params_)
#Guardamos el mejor modelo
Knn = knn_cv.best_estimator_
#Graficamos el desempeño obtenido para cada combinación de parámetros
grid_results = pd.DataFrame(knn_cv.cv_results_)
sn.catplot(x="param_knn_n_neighbors", y="mean_test_score", kind="point",
hue="param_knn__metric,data=grid_results)
```

```
Best params for k-neighborns:
{'knn_metric': 'euclidean', 'knn_n_neighbors': 3}
```

Figura 3: Desempeño de los mejores parámetros para k-vecinos




Como se observa en el gráfico anterior, el mejor desempeño (mayor acierto o *test score*) se obtiene con la distancia Euclidiana como medida de distancia entre vecinos y k=3.

## 2.4 *Random forest*

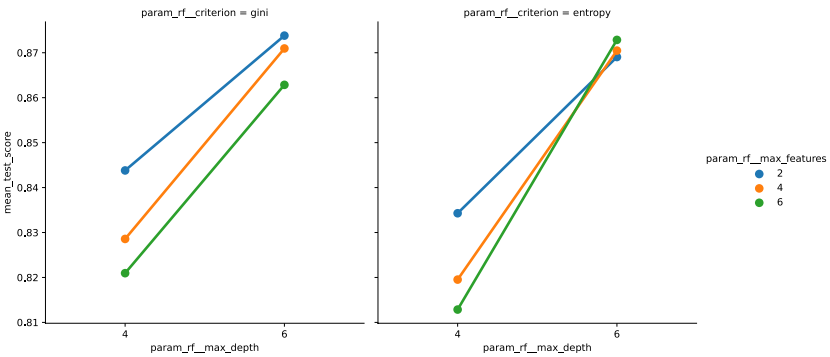
Los parámetros que se analizarán para *random forest* son los siguientes:

- *n_estimators:* cantidad de árboles a realizar. Lo dejamos fijo en 100.
- *max_features:* cantidad de variables que se deben considerar al buscar la mejor división.
- *max_depth:* profundidad máxima del árbol.
- *criterion:* función para medir la calidad de la división de cada árbol. Los dos criterios admitidos son la impureza de Gini (*gini*) y la entropía (*entropy*).

```
rf = RandomForestClassifier(n_estimators=100)
#Determinamos los parámetros a probar en la grilla
max_features= [2, 4, 6]
max_depth = [4,6]
criterion = ['gini', 'entropy']
parametros= dict(rf_max_features= max_features,
rf_max_depth = max_depth,
rf_criterion = criterion)
#Realizamos la grilla
model_rf = Pipeline(steps=[('sampling', SMOTE()), ('rf', rf)])
rf_cv=GridSearchCV(model_rf,parametros,cv=10)
rf_cv.fit(X_train_scaled, y_train_scaled)
#Vemos los mejores parámetros según la grilla
print('Best params for Random Forest:')
print(rf_cv.best_params_)
#Guardamos el mejor modelo
rf = rf_cv.best_estimator_
#Graficamos el desempeño obtenido para cada combinación de parámetros
grid_results = pd.DataFrame(rf_cv.cv_results_)
sn.catplot(x="param_rf_max_depth", y="mean_test_score", hue="param_rf_
max_features", kind="point", col="param_rf_criterion", data=grid_results)
```

```
Best params for Random Forest:
{'rf_criterion': 'gini', 'rf_max_depth': 6, 'rf_max_features': 2}
```

Figura 4: Desempeño de los mejores parámetros para Random Forest




Como se observa en los gráficos anteriores, el mejor desempeño (mayor acierto o *test score*) se obtiene con dos atributos seleccionados al azar para cada árbol, una profundidad de seis niveles y Gini como medida de «desorden» dentro de los nodos. Es muy interesante que el modelo prefiera tomar solo dos de los seis atributos para construir árboles. Esto es un reflejo de la naturaleza de los modelos de ensamblaje: se prefieren árboles «malos» con poca información pero que sean diferentes entre sí, de esta forma es posible tener diversas opiniones antes de combinarlas.

## *2.5 Red neuronal*

Definimos los parámetros a probar en la grilla:

- *batch_size:* tamaño de mini *batches* para el optimizador estocástico.
- *max_iter:* número máximo de iteraciones (número de épocas).
- *alpha:* parámetro de penalización (término de regularización L2).
- *hidden_layer_sizes:* número de neuronas en la capa oculta.

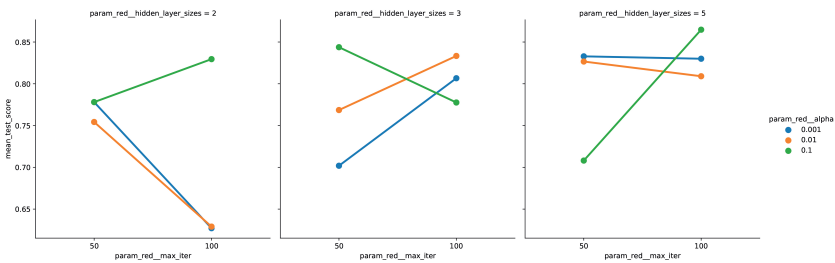
```
red = MLPClassifier(solver='lbfgs')
#Determinamos los parámetros a probar en la grilla
max_iter = [50,100]
alpha = [0.1, 0.01, 0.001]
```

```
hidden_layer_sizes = [2,3,5]
parametros= dict(red_max_iter = max_iter, red_alpha = alpha,
red_hidden_layer_sizes = hidden_layer_sizes)
#Realizamos la grilla
model_red = Pipeline(steps=[('sampling', SMOTE()), ('red', red)])
red_cv=GridSearchCV(model_red,parametros,n_jobs=2,cv=10)
red_cv.fit(X_train_scaled, y_train_scaled)
#Vemos los mejores parámetros según la grilla
print('Best params for neural network:')
print(red_cv.best_params_)
#Graficamos el desempeño obtenido para cada combinación de parámetros
red = red_cv.best_estimator_
grid_results = pd.DataFrame(red_cv.cv_results_)
sn.catplot(x="param_red_max_iter", y="mean_test_score", hue="param_red_
alpha", kind="point", col="param_red__hidden_layer_sizes",data=grid_results)
```

Best params for neural network:

{'red_alpha': 0.001, 'red_hidden_layer_sizes': 5, 'red_max_iter': 100}

Figura 5: Desempeño de los mejores parámetros para redes neuronales




Como se observa en los gráficos anteriores, el mejor desempeño (mayor acierto o *test score*) se obtiene con cinco neuronas en la capa oculta (tercer gráfico), un parámetro de regularización de 0.001 y 100 épocas.

## *2.6 Support vector machine*

Definamos los parámetros a probar en la grilla de SVM:

- *C:* parámetro de regularización L2. Controla el balance entre ajuste del modelo a los datos y la maximización del margen de separación de clases.
- *gamma:* coeficiente de *kernel*.

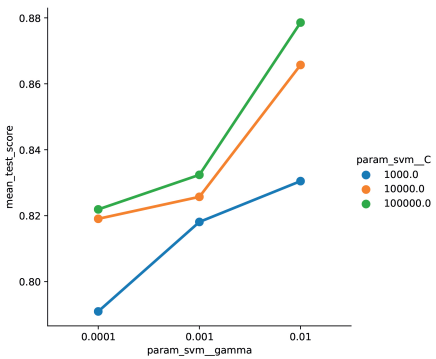
Se pueden encontrar más parámetros para este modelo en la documentación.

```
svm = SVC(probability=True)
#Determinamos los parámetros a probar en la grilla
C= [1e3, 1e4, 1e5]
gamma = [0.0001, 0.001, 0.01]
parametros= dict(svm_C= C, svm_gamma = gamma)
#Realizamos la grilla
model_svm = Pipeline(steps=[ ('sampling', SMOTE()), ('svm', svm)])
svm_cv=GridSearchCV(model_svm,parametros,cv=10) svm_cv.fit(X_train_scaled, y_train_scaled)
#Vemos los parámetros del mejor modelo
print('Best params for SVM:') print(svm_cv.best_params_)
#Guardamos el mejor modelo
svm = svm_cv.best_estimator_
#Graficamos el desempeño obtenido para cada combinación de parámetros
grid_results = pd.DataFrame(svm_cv.cv_results_)
sn.catplot(x="param_svm_gamma", y="mean_test_score", hue="param_svm_C",
kind="point" ,data=grid_results);
```

Best params for SVM:

{'svm_C': 100000.0, 'svm_gamma': 0.01}

Figura 6: Desempeño de los mejores parámetros para SVM



Como se observa en el gráfico anterior, el mejor desempeño (mayor acierto o *test score*) se obtiene con C=100000 y un parámetro de *kernel* gamma=0.01.

## 3. Evaluación

Evaluaremos el rendimiento de los modelos previamente entrenados para determinar su capacidad de predicción. Para esto utilizamos los datos de testeo. Primero creamos un *dataframe* vacío para ir guardando los resultados de predicción de cada modelo:

```
result_table = pd.DataFrame(columns=['classifiers', 'fpr','tpr','auc'])
```

### *3.1 Regresión logística*

Calculamos el *accuracy* del modelo con los datos de testeo:

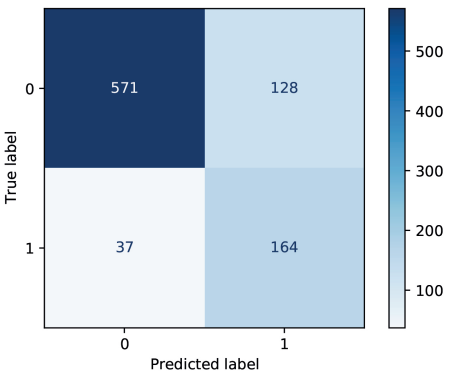
```
Ac_reg = regresion.score(X_test, y_test)
print('Acierto Regresión Logística: {:.2f}'.format(Ac_reg))
```

```
Acierto Regresión Logística: 0.82
```

Calculamos la matriz de confusión del modelo:

```
metrics.plot_confusion_matrix(regresion, X_test, y_test, values_format='.6g',
cmap='Blues')
plt.show()
```

Figura 7: Matriz de confusión del modelo



Calculamos el área bajo la curva ROC (AUC):

```
Y_pred_regresion = regresion.predict(X_test)
fpr, tpr, thresholds = roc_curve(y_test, regresion.predict_proba(X_test)[:,1])
 auc = roc_auc_score(y_test, Y_pred_regresion)
print(,AUC: %.3f‘ % auc)
```

AUC: 0.816

El desempeño de la regresión logística es bastante bueno (AUC>0.8), sobre todo en la predicción de los clientes buenos (ver matriz de confusión, *true label*=0).

Agregamos los resultados obtenidos para luego graficar la curvas ROC del modelo:

```
result_table = result_table.append({‘classifiers’: ‘Regresión logística’,’fpr’:fpr,
‘tpr’:tpr, ‘auc’:auc}, ignore_index=True)
```

Realizamos el mismo procedimiento con todos los modelos, como se muestra a continuación.

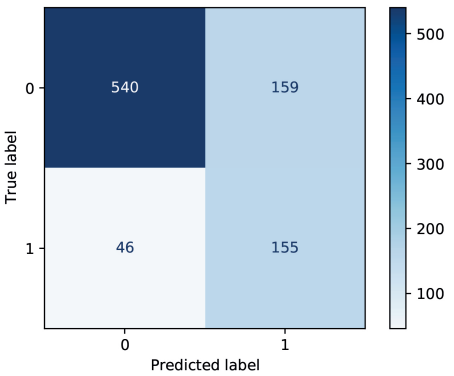
## *3.2 Árbol de decisión*

```
# Accuracy
Ac_tree = arbol.score(X_test, y_test) #con el modelo entrenado en entrenamiento
print(‘Acierto árbol de decisión: {:.2f}’.format(Ac_tree))
#matriz de confusión
metrics.plot_confusion_matrix(arbol, X_test, y_test, values_format=’.6g’,
cmap=’Blues ‘)
plt.show()
# Cálculo de AUC
Y_pred_tree = arbol.predict(X_test)
fpr, tpr, thresholds = roc_curve(y_test, arbol.predict_proba(X_test)[:,1])
auc = roc_auc_score(y_test, Y_pred_tree)
print(‘AUC: %.3f’ % auc)
#Agregar a la tabla de resultados
result_table = result_table.append({‘classifiers’: ‘Árbol de decisión’,’fpr’:fpr,
‘tpr’:tpr, ‘auc’:auc}, ignore_index=True)
```

Accuracy árbol de decisión: 0.77

AUC: 0.784

Figura 8: Matriz de confusión del modelo




Podemos observar que el desempeño del árbol es inferior al de la regresión logística. Esto es común cuando todas las variables predictoras son numéricas, como en este ejemplo.

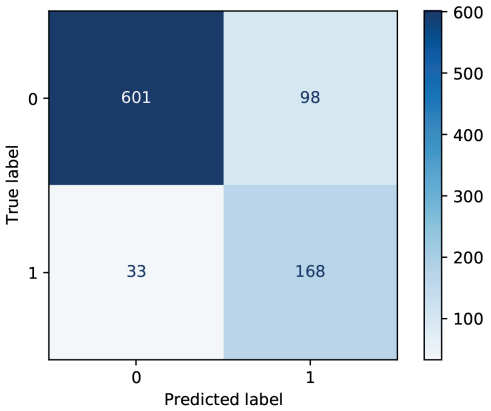
## *3.3 K-vecinos más cercanos*

```
# Accuracy
Ac_knn = Knn.score(X_test_scaled, y_test_scaled)
print('Acierto of k-vecinos más cercanos: {:.2f}'.format(Ac_knn))
#matriz de confusión
metrics.plot_confusion_matrix(Knn, X_test_scaled, y_test_scaled, values_
format='.6g',cmap='Blues')
plt.show()
# Cálculo de AUC
Y_pred_knn = Knn.predict(X_test_scaled)
fpr, tpr, thresholds = roc_curve(y_test_scaled, Knn.predict_proba(X_test_scaled)
[:,1])
auc = roc_auc_score(y_test_scaled, Y_pred_knn)
print('AUC: %.3f' % auc)
#Agregar a la tabla de resultados
result_table = result_table.append({'classifiers': ' K-vecinos más
cercanos','fpr':fpr, 'tpr':tpr, 'auc':auc}, ignore_index=True)
```

Accuracy k-vecinos más cercanos: 0.85

AUC: 0.836

Figura 9: Matriz de confusión del modelo



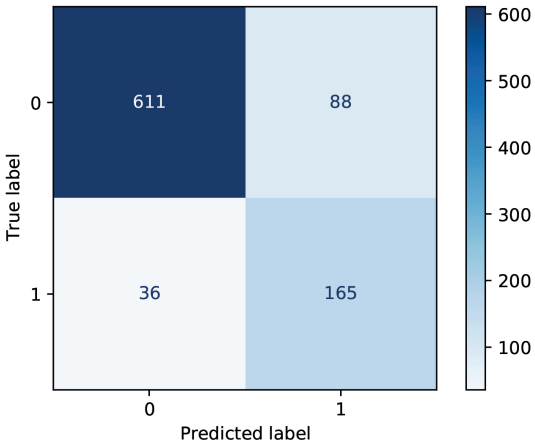

En este caso, k-vecinos más cercanos logran superar ligeramente el desempeño predictivo de la regresión logística.

## *3.4 Random Forest*

```
# Accuracy
Ac_rf = rf.score(X_test_scaled, y_test_scaled)
print('Acierto de random forest: {:.2f}'.format(Ac_rf))
#matriz de confusión
metrics.plot_confusion_matrix(rf, X_test_scaled, y_test_scaled, values_
format='.6g', cmap='Blues')
plt.show()
# Cálculo de AUC
Y_pred_rf = rf.predict(X_test_scaled)
fpr, tpr, thresholds = roc_curve(y_test_scaled, rf.predict_proba(X_test_scaled)
[:,1])
auc = roc_auc_score(y_test_scaled, Y_pred_rf)
print('AUC: %.3f' % auc)
#Agregar a la tabla de resultados
result_table = result_table.append({'classifiers': 'Random forest','fpr':fpr, 'tpr':tpr,
'auc':auc}, ignore_index=True)
```

Accuracy random forest: 0.86
AUC: 0.857

Figura 10: Matriz de confusión del modelo




*Random forest* logra superar a los métodos anteriores tanto en acierto global como en *accuracy*. Es notable que un método que se construye en base a árboles de decisión (que es el modelo de más bajo poder predictivo para estos datos) logre superar el resto de los modelos diseñados para datos numéricos.

## *3.5 Red neuronal*

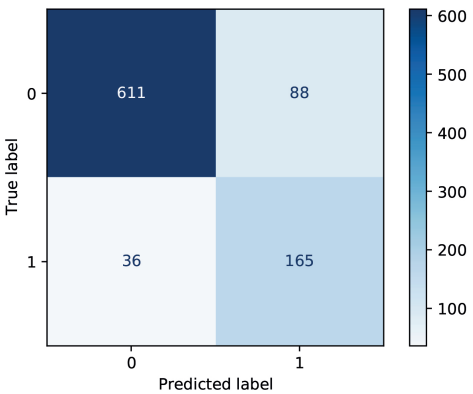
```
# Accuracy
Ac_red = red.score(X_test_scaled, y_test_scaled)
print('Acierto de red neuronal: {:.2f}'.format(Ac_red))
#matriz de confusión
metrics.plot_confusion_matrix(red, X_test_scaled, y_test_scaled, values_
format='.6g',cmap='Blues')
 plt.show()
# Cálculo de AUC
Y_pred_red = red.predict(X_test_scaled)
fpr, tpr, thresholds = roc_curve(y_test_scaled, red.predict_proba(X_test_scaled)
[:,1])
auc = roc_auc_score(y_test_scaled, Y_pred_red)
print('AUC: %.3f' % auc)
#Agregar a la tabla de resultados
```

```
result_table = result_table.append({'classifiers': 'Red neuronal','fpr':fpr, 'tpr':tpr, 'auc':auc}, ignore_index=True)
```

Acierto de red neuronal: 0.86

AUC: 0.885

Figura 11: Matriz de confusión del modelo



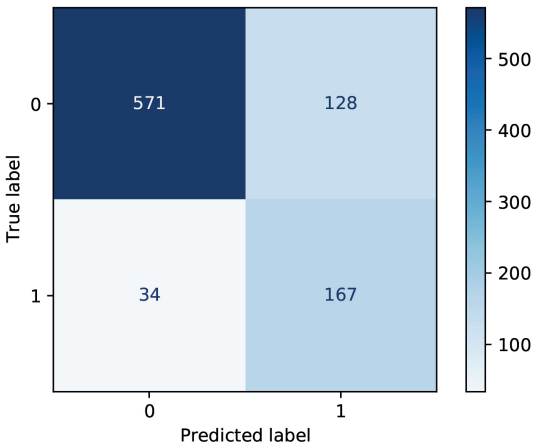

Las redes neuronales logran superar ligeramente a *random forest*. Constituyen, por ahora, el mejor modelo predictivo.

## *3.6 Support vector machine*

```
# Accuracy
Ac_svm = svm.score(X_test_scaled, y_test_scaled) #con el modelo entrenado en entrenamiento
print('Acierto de SVM: {:.2f}'.format(Ac_svm))
#matriz de confusión
metrics.plot_confusion_matrix(svm, X_test_scaled, y_test_scaled, values_format='.6g', plt.show()
# Cálculo de AUC
Y_pred_svm = svm.predict(X_test_scaled)
fpr, tpr, thresholds = roc_curve(y_test_scaled, svm.predict_proba(X_test_scaled)[:,1])
auc = roc_auc_score(y_test_scaled, Y_pred_svm)
print('AUC: %.3f' % auc)
#Agregar a la tabla de resultados
result_table = result_table.append({'classifiers': 'Support vector machine','fpr':fpr, 'tpr':tpr, 'auc':auc}, ignore_index=True)
```

Acierto de SVM: 0.88
AUC: 0.881

Figura 12: Matriz de confusión del modelo




Finalmente, el desempeño de SVM es el superior al de todos los otros métodos, alzándose como ganador en este «torneo» de modelos predictivos. SVM supera por bastante a los métodos interpretables (regresión logística y árbol de decisión), por lo que se justificaría la utilización de modelos de caja negra en caso de no haber restricciones regulatorias.

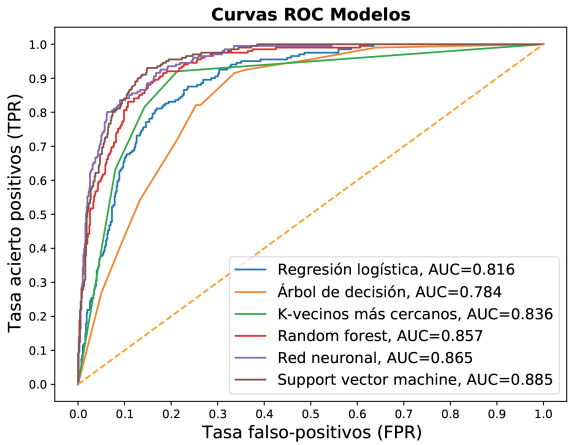
## *3.7 Curvas ROC*

A modo de resumen, graficamos las curvas ROC de todos los modelos:

```
result_table.set_index('classifiers', inplace=True)
fig = plt.figure(figsize=(8,6))
for i in result_table.index:
plt.plot(result_table.loc[i]['fpr'],
result_table.loc[i]['tpr'],
label="{}, AUC={:.3f}".format(i, result_table.loc[i]['auc']))
plt.plot([0,1], [0,1], color='orange', linestyle='--')
 plt.xticks(np.arange(0.0, 1.1, step=0.1))
```

```
plt.xlabel(«Tasa falso-positivos (FPR)», fontsize=15)
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel(«Tasa acierto positivos (TPR)», fontsize=15)
plt.title('Curvas ROC Modelos', fontweight='bold', fontsize=15)
 plt.legend(prop={'size':13}, loc='lower right')
plt.show()
```

Figura 13: Curvas ROC de todos los modelos




Como vemos, en los resultados obtenidos en la evaluación de los modelos, el que consigue mejores resultados es SVM. Guardamos este modelo entrenado en un archivo para luego, con él, realizar predicciones en el Laboratorio 4.

```
In [39]: %cd /content
drive.mount('/content/gdrive',force_remount=True )
dump(svm, 'gdrive/My Drive/modelo_entrenado.joblib')
```

# 9. *Predictive Analytics* IV: Regresión y series de tiempo

## 9.1 Introducción

En los capítulos anteriores revisamos modelos predictivos enfocados en clasificación, es decir, una tarea predictiva asociada a una variable nominal (identificar clientes «buenos» o «malos» en términos de devolver un crédito otorgado, o clientes que podrían fugarse de la compañía). Pero ¿qué pasa cuando lo que quiero no es etiquetar, o predecir una etiqueta, sino que quiero predecir en un problema que podría tomar cualquier valor continuo? Esta tarea se conoce como **regresión.**

El presente capítulo está dedicado a tareas predictivas en donde la predicción es una variable numérica, como, por ejemplo, ingresos, precios, balances de tarjeta de crédito, demanda, etcétera. Este tipo de modelos también forma parte de lo que se conoce como aprendizaje supervisado (ver capítulo 1), en donde tenemos un conjunto de datos que incluye el conocimiento *a priori* que se utiliza para el aprendizaje.

Supongamos que somos una empresa que produce cerveza y queremos determinar la demanda de tal preciado producto. Lo primero que haremos será recolectar información histórica de ventas de cerveza para construir tanto nuestra matriz **X** de predictores como nuestras variable objetivo Y, ahora numérica (unidades de cerveza vendidas). En base a **X** e Y construiremos un modelo predictivo, el cual aplicaremos a nuevos datos para conocer la demanda futura.

La estimación de demanda es un caso particular de modelo de regresión conocido como **series de tiempo**, en donde la matriz **X**

se construye en base a registros pasados de la misma variable que queremos predecir. Es decir, intentaremos conocer la demanda futura de este producto en base a la misma demanda, pero en el pasado. Es esperable que mi demanda por cerveza de mañana se parezca a la de hoy, a la de ayer y a la del mismo día de la semana pasada (no es lo mismo el consumo un viernes que un domingo). En palabras simples, una serie de tiempo corresponde a una sucesión de datos ordenados cronológicamente (Hyndman y Athanasopoulos, 2018).

Otro caso particular de modelos de regresión es el análisis de supervivencia *(survival analysis)*. Estos modelos se preocupan de predecir cuándo ocurrirá un determinado fenómeno. Por ejemplo, tareas de clasificación como *credit scoring* o fuga de clientes pueden adaptarse para responder las preguntas ¿cuándo un cliente caerá en *default*? o ¿cuándo un cliente dejará la compañía? La principal virtud del análisis de supervivencia es que los modelos son capaces de incorporar el hecho de que la respuesta a estas preguntas puede ser «nunca», es decir, permite dejar en claro cuando los clientes son «buenos pagadores» o «no fugados» en la modelación del problema de regresión. Dado que estos modelos son más bien de naturaleza econométrica, no serán discutidos en el libro.

Antes de abordar la modelación de series de tiempo, nos enfocaremos en adaptar los modelos predictivos vistos en los capítulos anteriores a la regresión. Algunos de ellos se pueden adaptar de forma bastante natural a esta tarea, mientras que otros requieren modificaciones un poco mayores. En este capítulo se discuten también las medidas de desempeño cuando la predicción es numérica, ya que difieren de las vistas para clasificación, como *accuracy* o AUC.

## 9.2 Modelos de regresión

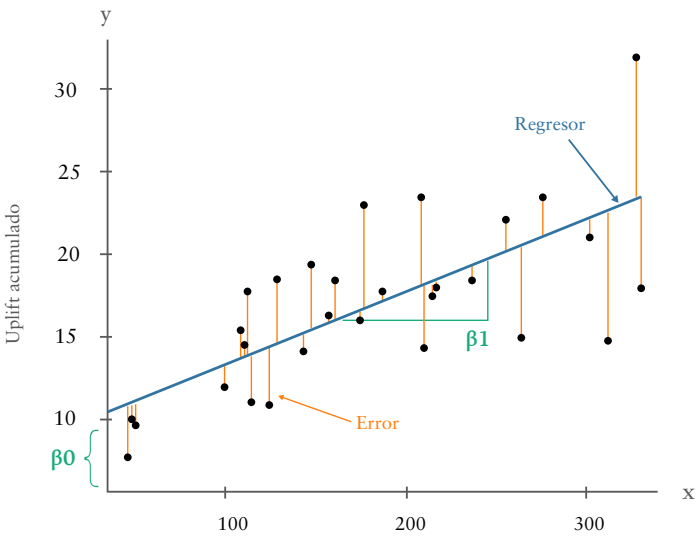
Comencemos con la interpretación geométrica de un modelo de regresión. A diferencia de un modelo de clasificación, ya no podemos representar las etiquetas con símbolos como círculos o cruces, por lo que necesitamos un eje dedicado a nuestra variable objetivo numérica. Ubicamos esta variable Y en la coordenada vertical, mientras

que en la abscisa dejamos una variable de la matriz **X.** El desafío ya no es separar dos o más conjuntos de etiquetas, sino construir una función que pase lo más cerca posible de todos los datos (Han *et al.*, 2011).

Si consideramos un hiperplano como el construido por el modelo de regresión lineal, ajustaremos las pendientes de tal forma de que no sobreestimemos ni subestimemos demasiado la variable Y. La regresión lineal, por ejemplo, calculará las distancias entre cada punto y el hiperplano, y minimizará estas distancias de tal manera que el hiperplano se encuentre lo más cerca posible de cada punto. Este proceso se conoce como Mínimos Cuadrados Ordinarios o MCO (OLS de acuerdo con las siglas en inglés).

La interpretación geométrica de una regresión lineal simple se puede ver en la figura 9.1. Llamaremos una regresión lineal simple a aquel modelo que contiene solo una variable explicativa y dos parámetros (una constante más la pendiente), mientras que un modelo con más de una variable explicativa se conoce como regresión múltiple.

Figura 9.1: Ejemplo de regresión lineal simple




Formalmente, MCO se basa en minimizar la medida *Mean Squared Error* (MSE). Cada punto negro en la ilustración corresponde al valor

real de la variable Y (por ejemplo, la demanda **real** por cerveza). La posición del hiperplano de donde nacen las líneas rojas corresponde a la predicción P (valores de demanda **predicha**). Para cada punto i, la diferencia entre Yi y Pi representan las líneas rojas, que pueden subir o bajar. Si sumamos los errores, estos se irán cancelando, dando la falsa sensación que el modelo es bueno. Debido a esto, se calculan las sumas entre estas dos variables al cuadrado:

Ecuación 9.1

$$MSE\ (Y,P) = \frac{1}{n}\sum_{i=n}^{n}(y_i - P_i)^2$$

El modelo de regresión lineal tiene una serie de supuestos asociados que pueden afectar negativamente el desempeño del modelo si no se cumplen. Por ejemplo, se espera que el término de error sea «ruido blanco», es decir, que la suma de los errores sea aproximadamente cero (es decir, que el hiperplano esté adecuadamente centrado en los datos) y sea independiente de las variables en **X** (James *et al.*, 2017).

La econometría se ha enfocado en realizar diversas adaptaciones al modelo si estos supuestos no se cumplen. Nosotros no entraremos en detalle en estas adaptaciones y asumiremos la regresión lineal como un modelo simple y que suele entregar buenos resultados, además de la ventaja de entregar la interpretación de sus coeficientes. Evaluaremos el modelo en modo predictivo y, si el desempeño no es bueno, nos inclinaremos por modelos que no hagan supuestos sobre los datos y que otorguen mayor flexibilidad, tales como *k* vecinos más cercanos (k-NN), *random forest* o redes neuronales.

Pensemos ahora en los modelos discutidos en los capítulos anteriores y su extensión a regresión (Shmueli *et al.*, 2010; James *et al.*, 2017):

- El método k-NN se puede adaptar de forma relativamente simple a regresión. Dado que la definición del vecindario se realiza calculando distancias dentro de la matriz X, no afecta en nada que Y sea ahora numérica. La modificación se hace en el paso

siguiente: k-NN para clasificación asigna una nueva instancia a la clase más representada, es decir, toma la **moda** de las clases en el vecindario. En regresión, asignaremos un valor numérico a la nueva instancia en base a la **media** de las observaciones en el vecindario.

- Para adaptar árboles de decisión a regresión se sigue la misma idea: en clasificación, se toma la moda de las etiquetas en cada hoja para asignar la predicción de un dato nuevo. En este caso tomaremos la media de los valores Y asociados a los datos en cada hoja. Sin embargo, medidas como entropía o Gini ya no son válidas para elegir el atributo a ramificar, pues se basan en una etiqueta nominal. Se utiliza, en cambio, la medida MSE (entre otras alternativas), eligiendo aquella variable cuya ramificación implique un menor error desde el punto de vista numérico.
- Habiendo adaptado los árboles de decisión a regresión, el ajuste de *random forest* es trivial: combinamos una serie de árboles para regresión, los cuales se combinan tomando la media de cada una de las predicciones hecha por cada árbol (en vez de la moda o votación por mayoría).
- Las redes neuronales, al igual que la regresión lineal, son modelos basados en funciones y, por ende, son más naturales en su forma de regresión que en clasificación. Esto debido a que tanto el lado izquierdo como el derecho de la ecuación $Y = f(X)$ es numérico. En el capítulo anterior se discutieron diferentes funciones de pérdida para las redes neuronales, tales como *cross-entropy,* que es la extensión a multiclase de la función *logit*. En el caso de regresión, no requerimos una función de enlace de este tipo que haga conversar un lado izquierdo nominal con un lado derecho numérico, y podemos utilizar una función como MSE.
- Los modelos Máquinas de Soporte Vectorial (SVM) y Naive Bayes fueron diseñados para clasificación, y su adaptación a regresión requiere modificaciones algo más sofisticadas. Estas adaptaciones no serán discutidas en este libro.

Finalmente, existen otras adaptaciones necesarias en regresión dentro del proceso de extracción de conocimiento. El preprocesamiento de datos es prácticamente similar en clasificación y regresión, salvo por los filtros de selección de atributos y, en particular, la evaluación de relevancia. En el capítulo 2 discutimos la utilización del *test* Chi Cuadrado como medida para la relevancia, la cual relaciona dos variables nominales. En regresión, en cambio, la variable objetivo es continua y, por ello, una medida como la correlación de Pearson puede ser mejor para evaluar independencia entre las variables de la matriz **X** y el vector Y.

Otro cambio menor ocurre en la validación de modelos. Hablamos en el capítulo 6 de que la definición de particiones para validación simple *(holdout)* o validación cruzada debe ser estratificada, es decir, que todas las clases deben estar representadas en la misma proporción en los diferentes conjuntos de datos construidos. Esto no es necesario en regresión, es decir, no existe el problema de desbalance de clases.

## 9.3 Medidas de desempeño para salida continua

Al igual como vimos las principales medidas de desempeño para salida discreta (clasificación) e introdujimos la matriz de confusión, en el caso de la salida continua tendremos otras medidas de desempeño (adicionales a MSE) que nos ayudarán a validar nuestros modelos.

Como ya definimos, la elección del mejor modelo depende de los vectores $Y_i$ y $P_i$; el valor real numérico y la predicción para cada dato $i$. MSE suele ser la medida elegida para la calibración o entrenamiento de muchos modelos de regresión, sin embargo, no suele ser seleccionada como la medida para tareas de validación o testeo.

Uno de los problemas de MSE es que tiene unidades de medida al cuadrado con respecto a la variable objetivo, lo que la hace poco intuitiva. Si buscamos predecir ventas de cerveza, ¿qué sentido tiene un error promedio de 1.000.000 de cervezas al cuadrado? Además, elevar a una potencia puede resultar en números muy grandes.

Una forma de solucionar este problema es elegir medidas que utilicen las mismas unidades que la variable Y. En este caso, si el modelo tiene un error promedio de 1000 cervezas, entonces podremos tener una noción más clara si este error en la estimación de demanda es alto o bajo. Las dos medidas de regresión más populares son probablemente *Mean Absolute Error* (MAE) y *Root Mean Squared Error* (RMSE), las cuales tienen esta propiedad:

Ecuación 9.2

$$MAE\ (Y,P) = \frac{1}{n}\sum_{i=n}^{n} |y_i - P_i|$$

Ecuación 9.3

$$RMSE\ (Y,P) = \sqrt{\frac{1}{n}\sum_{i=n}^{n} (y_i - P_i)^2}$$

Como se puede ver en la ecuación 9.2, la medida MAE se construye reemplazando las diferencias al cuadrado por las diferencias en valor absoluto. De esta forma se llega a las mismas unidades que Y. La medida RMSE en ecuación 9.3, en cambio, se obtiene aplicando la raíz cuadrada a MSE (ecuación 9.1).

Finalmente, existen medidas adimensionales que pueden resultar más intuitivas que las anteriores. Esto gracias a que una medida adimensional no depende del problema a abordar. Por ejemplo, un error promedio de un 2% parece bastante bajo independiente de la tarea predictiva, con relación a un error de un modelo que podría ser de un 30%. La medida *Mean Absolute Percentage Error* (MAPE) obtiene justamente un error porcentual promedio al dividir las diferencias entre $Y_i$ y $P_i$ por $Y_i$, el valor real:

Ecuación 9.4

$$MAPE\ (Y,P) = \frac{1}{n}\sum_{i=n}^{n} \left| \frac{y_i - P_i}{y_i} \right|$$

Una acotación importante sobre MAPE: esta medida se puede indefinir si existen valores de Y iguales a cero. Por ejemplo, si hay periodos en donde no se vendió cerveza, el valor de MAPE arrojará un error. Una forma simple de corregirlo es omitir los ceros en el conjunto de validación o testeo. Otra medida adimensional es la correlación de Pearson entre Y y P. A diferencia de las medidas de error anteriores, una mayor correlación implica un mejor ajuste de modelo.

## 9.4 Modelos de serie de tiempo

Como se indicó en la introducción, una serie de tiempo corresponde a una secuencia de observaciones únicas, que han sido observadas en instantes sucesivos en el tiempo. Al igual que otros modelos predictivos, puede resultar muy útil el comprender la estructura intrínseca de la serie de tiempo antes de realizar predicciones, del mismo modo que hicimos *descriptive analytics* antes de correr modelos.

Para comenzar con el análisis de series de tiempo, debemos comprender cuáles son los principales componentes que pueden manifestarse en las series. Para esto introduciremos cuatro componentes: la tendencia, la estacionalidad, la variación cíclica y el ruido.

La **tendencia** expresa la progresión a largo plazo de la serie temporal. Es decir, nos ayuda a verificar si en el largo plazo la serie tiende a aumentar su valor, o bien a disminuirlo.

Por otro lado, la **estacionalidad** es el componente que nos permite estudiar fluctuaciones sistemáticas que ocurren dentro de un periodo corto de tiempo. Básicamente corresponde a patrones que se repiten a lo largo de la serie para cada periodo, ya sean diarios, semanales, mensuales, anuales, etcétera.

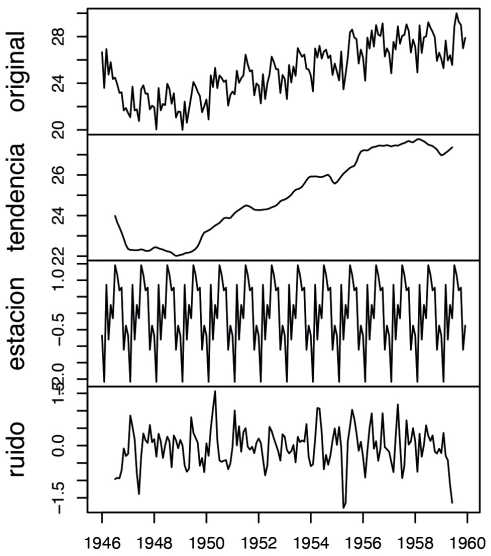
Entender las diferentes estacionalidades de una serie de tiempo es clave para el modelador, ya que este debe indicarlas en el proceso de estimación. No es fácil para la máquina identificar las estacionalidades de forma automática, de ahí que un modelo que no reconozca estacionalidades es muy probable que fracase estrepitosamente en términos de desempeño predictivo.

La **ciclicidad** de las series corresponde a la fluctuación que se puede apreciar en las series con un carácter cíclico, normalmente mayor a dos años. Esto nos puede servir para apreciar aumentos o caídas que se escapan de la componente estacional.

Finalmente, tenemos el **ruido** de la serie. Corresponde a todo lo que no podemos explicar de la serie una vez estudiada la tendencia, estacionalidad y ciclos. Podría ser catalogado como los residuos de la serie.

Una forma de realizar *descriptive analytics* en series de tiempo es mediante la **descomposición estacional.** Esto consiste en una representación gráfica de la serie, donde en la abscisa se ubica el tiempo, mientras que en el eje Y se ubica el valor de la serie. Luego se descompone la serie original en tendencia, estacionalidad y ruido. En la siguiente imagen vemos cómo se puede observar gráficamente la descomposición de una serie:

Figura 9.2: Descomposición estacional de una serie de tiempo




En la figura 9.2 se muestra la serie original (primer gráfico), la cual se descompone en las tres series siguientes: la tendencia (es relativamente irregular, pero ligeramente creciente), el patrón estacional (repetitivo dentro de un año), y finalmente el ruido, que incluye todas las fluctuaciones de corto plazo que no pueden ser explicadas por la estacionalidad.

Revisemos ahora algunos de los modelos clásicos para la predicción de modelos de serie de tiempo, para luego discutir algunas adaptaciones de modelos de aprendizaje de máquinas.

### *9.4.1 Modelo Holt-Winters (suavizamiento exponencial)*

El modelo suavizamiento exponencial fue propuesto a finales de los años cincuenta y ha motivado algunos de los métodos de pronóstico más utilizados. Los pronósticos que realizan estos modelos son sumas ponderadas de los ejemplos pasados, donde los pesos o ponderaciones decaen exponencialmente a medida que las observaciones son más antiguas.

Para estudiar estos modelos definiremos la siguiente nomenclatura:

- $Y_t$ la observación en el periodo t.
- $P_{t+1}$ el pronóstico para el periodo T+1.
- $A_t$ el promedio móvil de las $N$ observaciones anteriores a t (incluyéndolo).
- $e_t$ el error o residuo.
- $T_t$ la tendencia.
- $C_t$ la estacionalidad de la serie.

El primer método que revisaremos es la suavización exponencial simple. Podemos definir como $A_t$ el promedio móvil de las $N$ observaciones más recientes como:

Ecuación 9.5

$$A_t = \frac{Y_t + Y_{t-1} + \cdots + Y_{t-N+1}}{N}$$

La suavización exponencial simple toma en consideración el promedio móvil de las observaciones anteriores y la última observación, pudiendo expresarlo como:

Ecuación 9.6

$$A_t = \alpha Y_t + (1-\alpha)A_{t-1}, \qquad \alpha \in [0,1]$$

La predicción para la observación t+1 viene dada por $P_{t+1} = A_t$, con $\alpha$ siendo el peso que se le asigna a la última observación por sobre el promedio móvil de la serie. Este parámetro puede ser arbitrario (por ejemplo, 0.5), o bien ser calibrado en base al mejor ajuste a los datos.

Este modelo tiene un problema: no es capaz de incorporar tendencia ni estacionalidad, por ende, solo es capaz de aprender de observaciones recientes. Como se destacó anteriormente, muchas aplicaciones tienen patrones estacionales, por lo que este tipo de modelos no llegará a buenos resultados. Afortunadamente, el modelo completo de Holt-Winters, también conocido como suavizamiento

exponencial triple, incorpora tanto estacionalidad como tendencia. Toma la siguiente forma:

Ecuación 9.7

$$A_t = \alpha \frac{Y_t}{C_{t-L}} + (1-\alpha)(A_{t-1} - T_{t-1})$$
$$T_t = \beta(A_t - A_{t-1}) + (1-\beta)T_{t-1}$$
$$C_t = \gamma \frac{Y_t}{A_t} + (1-\gamma)C_{t-L}$$
$$P_{t+k} = (A_t + kT_t)C_{t-L+((k-1) mod\ L)}$$

En este caso, el pronóstico se realiza en base a cuatro ecuaciones e incluye tres parámetros ajustables que varían entre 0 y 1: $\alpha$, $\beta$ y $\gamma$. Estos parámetros balancean la importancia que le da al modelo a la última observación, la tendencia y la estacionalidad, respectivamente (Hyndman y Athanasopoulos, 2018).

Explicar las cuatro ecuaciones resulta complejo y no es realmente necesario, pues lo importante es indicar que $L$ representa el periodo de la estacionalidad (cantidad de observaciones que hay que contar hacia atrás para encontrar un patrón cíclico) y debe ser explicitado por el modelador. Algunos ejemplos de estacionalidades comunes se mencionan a continuación:

- Si los datos (por ejemplo, ventas de cervezas) son mensuales, se sugiere explicitar $L = 12$ con el fin de identificar patrones anuales. Por ejemplo, se puede vender más cerveza en verano que en invierno.
- Si los datos son diarios, se sugiere explicitar $L = 7$ con el fin de identificar patrones semanales. Por ejemplo, se puede vender más cerveza los viernes y sábado, con relación al resto de la semana.
- Si los datos son por hora, se puede explicitar $L = 24$ con el fin de identificar patrones diarios. Por ejemplo, se puede vender más cerveza hacia el final del día más que en la mañana.

### *9.4.2 Modelo Box-Jenkins (ARIMA)*

Además de Holt-Winters, otro modelo muy utilizado es ARIMA, parte de la familia de modelos de Box-Jenkins. Estos modelos consideran que existe una dependencia entre las observaciones, es decir, cualquier observación puede ser explicada como una función de las anteriores. Los modelos de Box-Jenkins son extensiones de la regresión lineal, en el sentido que se basan en la minimización del término de error para encontrar los coeficientes o ponderadores de la información pasada.

Dentro de los modelos Box-Jenkins, existen algunos más simples y otros más complejos que resultan de la combinación de modelos simples. Veamos los principales modelos Box-Jenkins:

- **Modelos autorregresivos *AR(p)*:** el pronóstico resulta de una combinación de las observaciones más recientes, cada una ponderada por un peso lineal $\alpha$ y una componente de error aleatorio $e_t$. El parámetro p define la cantidad de **rezagos** u observaciones recientes que se consideran. Por ejemplo, p=2 quiere decir que mis ventas de cerveza de hoy ($P_t$) corresponden a una suma ponderada entre las ventas de ayer y las de anteayer ($\alpha_1 Y_{t-1} + \alpha_2 Y_{t-2}$ ). Los valores de $\alpha_1$ y $\alpha_2$ se obtienen mediante MCO, minimizando la suma al cuadrado de los residuos.

Ecuación 9.8

$$P_t = \sum_{i=1}^{p} \alpha_i Y_{t-i} + e_t$$

- **Modelos de media móvil MA(q):** en vez de utilizar los valores pasados de la variable a predecir, estos modelos consideran los errores de pronóstico pasados como una suma ponderada. Dado que estos errores no son observables, el modelo se vuelve menos intuitivo. Sin embargo, la incorporación de residuos pasados puede llevar a un mejor pronóstico. Similar a los modelos AR,

el parámetro $q$ define la cantidad de residuos recientes que se consideran.

Ecuación 9.9

$$P_t = \sum_{j=1}^{N} \beta_j e_{t-j} + e_t$$

- **Modelo combinados ARMA(p,q):** corresponde a una combinación lineal de los dos modelos AR y MA. Considera en la modelación tanto el pasado reciente como los errores de pronóstico previos.

Ecuación 9.10

$$P_t = \sum_{i=1}^{p} \alpha_i Y_{t-i} + \sum_{j=1}^{q} \beta_j e_{t-j} + e_t$$

- **Modelos ARIMA(p,d,q) y concepto de integración:** es más fácil estimar un modelo ARMA cuando la serie es estacionaria (¡no confundir con el concepto de estacionalidad!), es decir, cuando no sigue una tendencia creciente. Construir modelos ARMA sobre una serie estacionaria permite obtener coeficientes más estables, ayudando a una mejor predicción. ARIMA es un acrónimo para *Auto-Regressive Integrated Moving Average*. En este contexto, la «I» en ARIMA se refiere al proceso de integración, que es el inverso de diferenciación. La diferenciación consiste en definir una nueva serie $Z_t$ restando los valores originales por la misma variable rezagada en un periodo: $Z_t = Y_t - Y_{t-1}$. Este proceso se puede repetir $d$ veces, que es el parámetro de este modelo adicional a $p$ y $q$. Con cada diferenciación «aplanamos» la tendencia y nos acercamos a tener una serie estacionaria. Cuando conseguimos esta estacionariedad, corremos un modelo ARMA con la nueva serie.

- **Modelo ARIMA estacional SARIMA(p,d,q)(sp,sd,sq)[L]:** anteriormente destacamos lo importante de modelar la estacionalidad y no solamente considerar datos recientes. SARIMA permite hacer esto y lo hace combinando dos modelos ARIMA: uno para los datos recientes y otro para un proceso estacional que va considerando rezagos de L en L periodos. Para esto se requieren dos conjuntos de parámetros: (p,d,q) para el proceso ARMA de datos recientes y (sp,sd,sq) para el proceso estacional.
- **Auto-ARIMA:** calibrar los seis parámetros (p,d,q,sp,sd,sq) del modelo SARIMA parece un desafío mayor de prueba y error para llegar a una buena configuración. Afortunadamente, el modelo Auto-ARIMA propuesto por Hyndman y Khandakar (2008) permite encontrar los parámetros óptimos de forma automática, evitando un tedioso proceso de búsqueda.
- **Modelo ARIMA estacional con variables exógenas SARIMAX(p,d,q)(sp,sd,sq)[L]:** el modelo de Box-Jenkins más completo considera además la opción de incluir variables exógenas (la «X» en SARIMAX viene de *eXogenous*). Estas variables contienen información diferente a la de la serie. Volviendo a nuestro ejemplo de demanda de cerveza, una variable exógena puede ser el pronóstico de la temperatura para el día a predecir (uno podría asumir que se consume más cerveza si el día es más caluroso) o la información de feriados o eventos especiales (fiestas nacionales, partidos de la selección de fútbol, etcétera).

Es importante mencionar que existen adaptaciones de modelos de aprendizaje de máquinas para series de tiempo. En general, es bastante directo: se construye la matriz **X** en base a varios rezagos de la serie de tiempo y se entrena un modelo de regresión. La matriz **X** suele contener rezagos recientes y estacionalidades con el fin de emular la información autorregresiva que entra en un modelo SARIMA. Adicionalmente, se pueden incorporar variables exógenas. Métodos como redes neuronales o SVM han sido ampliamente utilizados para series de tiempo.

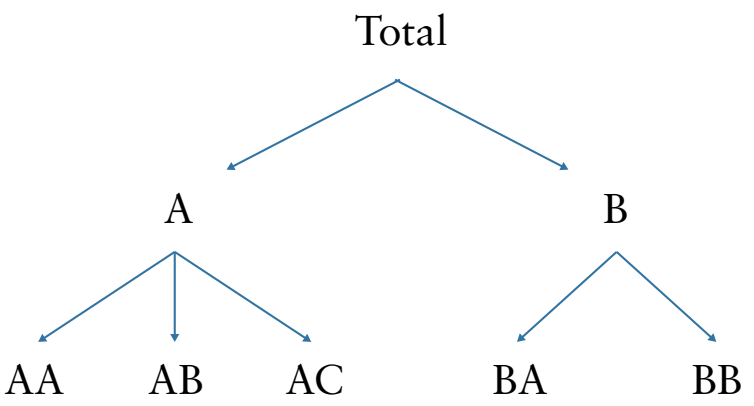
### *9.4.3 Series de tiempo jerárquicas*

Hemos estudiado las series de tiempo y sus principales componentes. Sin embargo, ¿podemos obtener aún más información de una serie de tiempo? ¿Existe alguna información «oculta» que podríamos utilizar para mejorar los pronósticos? La respuesta es que, en muchos casos, sí existe.

Muchas veces las series de tiempo responden implícitamente a una estructura jerárquica. Supongamos que tenemos una cadena de tiendas de ropa. Vendemos diversos artículos de vestir en nuestras tiendas. Una serie de tiempo muy interesante puede ser la venta total de nuestra empresa. Podemos estudiar la cantidad de prendas o los ingresos que hemos percibido en el tiempo. Sin embargo, detrás de estas series existen muchas otras series de tiempo que puede ser interesante estudiar y que de alguna forma «nacen» de esta serie principal. Estas pueden ser, por ejemplo, la venta de algún producto específico (nivel SKU) o de una categoría de ropa en particular (camisas, por ejemplo).

Supongamos entonces que nuestra venta total la desagregamos en estos dos niveles jerárquicos. Primero la demanda por categorías de producto, luego por SKU. Cada una de esas series individuales explica en cierto modo la serie total. Para ejemplificar esto, podemos utilizar el siguiente árbol:

Figura 9.3: Estructura jerárquica de las series de tiempo



En este ejemplo se desagrega la venta total en la tienda de ropa por dos categorías: camisas (categoría A) y pantalones (categoría B). Estas categorías se desagregan nuevamente en diferentes SKU por marca de ropa: los productos AA, AB, AC, BA y BB. Si sumamos cualquier nivel jerárquico, tendremos la venta total.

La principal motivación de las series de tiempo jerárquicas es que la demanda por producto (nivel más bajo de la jerarquía) suele contener mucho ruido, ya que las ventas pueden ser muy pocas en cada periodo. Además, los SKU cambian, o los productos se sustituyen por parte del cliente en caso de falta de *stock*, ensuciando los datos. Para contrarrestar este efecto, podemos estimar las series de tiempo en una categoría intermedia, para luego proyectar lo que pasaría en los nodos hoja (Karmy y Maldonado, 2019).

Volviendo a nuestro ejemplo, puede tener más sentido estimar la demanda por camisas en la tienda, donde seguramente se encontrarán claros patrones estacionales, para luego desagregar esta estimación en las proporciones que corresponden a cada marca y talla. Finalmente, las ventas totales se obtienen sumando los pronósticos por camisas y pantalones. Este enfoque se conoce como *middle-out,* ya que construye modelos en una categoría intermedia y calcula las series restantes «de adentro hacia afuera».

Alternativamente, el enfoque *bottom-up* consiste en construir un modelo para cada uno de los nodos del último nivel de la jerarquía, es decir, para los datos más desagregados (nivel de SKU en nuestro ejemplo). Las predicciones de los niveles superiores se realizan sumando las series que los conforman de acuerdo con la estructura del árbol jerárquico.

Si definimos $Y_t$ como la serie total, y $Y_{n,t}$ como la serie la asociada al nodo $n$, las ecuaciones que permiten llegar a los pronósticos de *bottom-up* para nuestro ejemplo de ventas de ropa son las siguientes:

Ecuación 9.11

$$\hat{y}_{A,h} = \hat{y}_{AA,h} + \hat{y}_{AB,h} + \hat{y}_{AC,h}$$
$$\hat{y}_{B,h} = \hat{y}_{BA,h} + \hat{y}_{BB,h}$$
$$\hat{y}_{h} = \hat{y}_{A,h} + \hat{y}_{B,h}$$

Este método tiene la gran ventaja de no perder información debido a las agregaciones. Eso no quita que, en algunos casos, los datos del último nivel incrementen el «ruido» de las series agregadas, lo que podría afectar en la predicción de manera negativa.

El enfoque *top-down* consiste en todo lo contrario. Se realiza la predicción de la serie total $y_t$ para luego desagregar dicha predicción hacia los niveles inferiores utilizando alguna proporción predeterminada. Existen diversas medidas que podemos utilizar para esta desagregación, siendo la más utilizada el promedio histórico de dicho nodo:

Ecuación 9.12

$$p_j = \frac{1}{T}\sum_{j=1}^{T}\frac{y_{j,T}}{y_t}$$

En el ejemplo anterior, las predicciones quedarían como sigue:

Ecuación 9.13

$$\hat{y}_{A,h} = p_A + \hat{y}_h, \hat{y}_{B,h} = p_B + \hat{y}_h$$
$$\hat{y}_{AA,h} = p_{AA} + \hat{y}_{A,h}, \hat{y}_{AB,h} = p_{AB} + \hat{y}_{A,h}, \hat{y}_{AC,h} = p_{AC} + \hat{y}_{A,h}$$
$$\hat{y}_{BA,h} = p_{BA} + \hat{y}_{B,h}, \hat{y}_{BB,h} = p_{BB} + \hat{y}_{B,h}$$

Una desventaja de este método es la potencial pérdida de información a la hora de realizar las desagregaciones, pérdida que puede incrementar con cada desagregación adicional. En nuestros estudios, los modelos con mejores resultados empíricos han sido *bottom-up* y *middle-out*.

Como el lector ya probablemente logró deducir, *middle-out* es una mezcla de los dos modelos anteriores. Se define un nivel inicial distinto al total y al último nivel, en el cual se efectúan las

predicciones, y luego se utiliza ambos enfoques: *top-down* para estimar las predicciones en los nodos inferiores en base a las proporciones, y *bottom-up* para estimar niveles superiores mediante sumas (Hyndman y Athanasopoulos, 2018).

### *9.4.3.1 Validación en series de tiempo*

Ya habiendo definido tanto las principales medidas de desempeño como los modelos más utilizados, este capítulo finaliza con un comentario importante sobre validación en series de tiempo: las particiones para entrenamiento y validación/testeo nunca deben ser realizadas al azar (Hyndman y Athanasopoulos, 2018).

Independientemente del modelo predictivo, una serie de tiempo considera rezagos de la variable Y para la construcción de la matriz **X**. Si particionamos los datos al azar, es bastante seguro que testeamos nuestro modelo con datos que ya utilizamos en el conjunto de entrenamiento, violando la regla más importante para una validación de modelos correcta e insesgada.

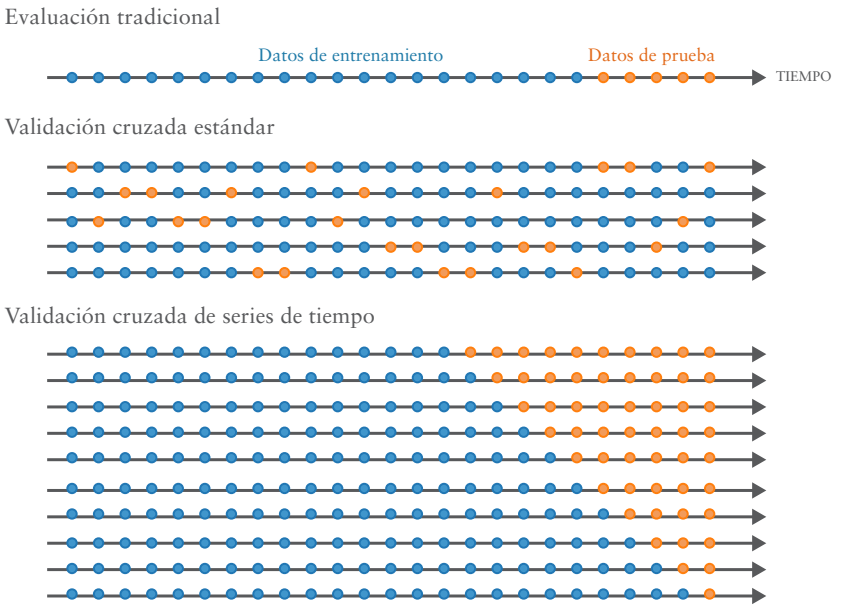
En series de tiempo se realizan validaciones *out-of-time* o fuera de tiempo, es decir, el conjunto de validación o testeo debe ser siempre posterior en tiempo al de entrenamiento. Una validación simple de tres etapas (un conjunto de validación para calibrar parámetros y uno de testeo para la comparación final entre modelos) quedaría de la siguiente manera (ver figura 9.4):

Figura 9.4: Validación de tres conjuntos para series de tiempo

El método de validación cruzada también se puede extender a series de tiempo. Hyndman, en el año 2012, introduce el concepto de *rolling forecasting origin*, que consiste en validar utilizando el orden cronológico de los datos e ir tomando subconjuntos de datos

consecutivos, como se puede apreciar en la figura 9.5 (Hyndman y Athanasopoulos, 2018):

Figura 9.5: Validación cruzada para series de tiempo




Pensemos en nuestro ejemplo de ventas de cerveza. Asumiendo datos diarios para un año completo (digamos, 2021), una estrategia de validación simple sería:

- Seleccionamos los primeros 10 meses del año para entrenamiento, utilizando noviembre de 2021 para seleccionar los mejores parámetros para cada uno de los modelos. De esta forma tengo 30 predicciones (los días del mes de noviembre), que podemos promediar y luego seleccionar aquella configuración de parámetros que lleva a un menor error.
- Seleccionamos las mejores configuraciones para cada modelo y los volvemos a entrenar con los datos de los primeros 11 meses del año. Ahora utilizamos los 31 días de diciembre para hacer una comparación entre diferentes modelos de series de tiempo

(sin recalibrar parámetros) y elegimos el mejor. Este lo utilizaremos para pronosticar nuevas ventas en 2022.

Validación cruzada, en cambio, quedaría de la siguiente manera:

- Definimos k=5, es decir, cinco particiones para promediar, construyendo cinco modelos.
- (K=1) Entrenamos con los primeros cinco meses del año, utilizando los cinco siguientes para seleccionar los mejores parámetros para cada uno de los modelos. Guardo el error promedio para K=1 para cada configuración de parámetros y cada modelo.
- (K=2) Entrenamos con los primeros seis meses del año, utilizando los cuatro siguientes para seleccionar los mejores parámetros para cada uno de los modelos. Guardo el error promedio para K=2 para cada configuración de parámetros y cada modelo.
- Repito el proceso de agregar un nuevo mes hasta K=5. Promediamos los cinco errores promedio obtenidos y seleccionamos aquella configuración de parámetros que lleva a un menor error.
- El proceso de testeo es similar al anterior: entrenamos con los datos de los primeros 11 meses del año cada modelo utilizando la mejor configuración de parámetros y lo evaluamos con diciembre de 2021. Elegimos el modelo de mejor desempeño para pronosticar nuevas ventas en 2022.

Como se indicó anteriormente para clasificación, la validación cruzada es más costosa en lo computacional, ya que implica calibrar muchos más modelos que con una validación simple. Sin embargo, nos permite tomar decisiones más robustas al momento de elegir el mejor modelo a utilizar (Hyndman y Athanasopoulos, 2018; Karmy y Maldonado, 2019).

## Referencias

Han, J., Kamber, M. & Pei, J. (2011). *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann Publishers.

Hyndman, R. J. & Khandakar, Y. (2008). «Automatic time series forecasting: the forecast package for r». *Journal of Statistical Software*, 27(3), pp. 1-22.

Hyndman R. J. & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. Melbourne: O Texts.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Karmy, J. P. & Maldonado, S. (2019). «Hierarchical Time Series Forecasting via Support Vector Regression in the European Travel Retail Industry». *Expert Systems and Applications,* 137, 59-73.

Shmueli, G., Patel, N. R. & Bruce, P. C. (2010). *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. New York: Wiley.

# 10. *Prescriptive Analytics*

El análisis de datos y la modelación predictiva deben ir de la mano con la toma de decisiones en el mundo de los negocios. La analítica prescriptiva es la última de las capacidades analíticas y pone en acción todos los elementos discutidos en este libro, más la información cuantitativa, que permite optimizar y automatizar las decisiones empresariales.

El *prescriptive analytics,* sin embargo, solo es efectivo en organizaciones con un nivel de madurez de analítica avanzado. Si los *inputs* y supuestos que se realizan no son válidos, los resultados no serán precisos. En este sentido, sigue el mismo principio GIGO indicado anteriormente para la modelación predictiva: *garbage in, garbage out*. Este fenómeno, sin embargo, se puede agravar si se combinan diferentes fuentes de datos y modelos.

Utilizada de forma efectiva, la analítica prescriptiva puede ayudar a empresas y organizaciones a tomar decisiones basadas en hechos y «datos duros», en vez de saltar a conclusiones apresuradas basadas en el instinto.

## 10.1 Optimización y simulación

La analítica prescriptiva tiene el potencial de evaluar de forma óptima diferentes desenlaces y escenarios con el fin de llegar a la mejor decisión. Adicionalmente, puede calcular la probabilidad de estos escenarios, para así evaluar el nivel de riesgo. Las empresas que emplean estas técnicas logran una comprensión mayor de los efectos de las decisiones bajo diferentes escenarios de incertidumbre, eligiendo actuar de forma acorde a sus directrices. La planificación

puede ser en base a promedios, pero también considerando decisiones robustas que funcionen adecuadamente bajo escenarios pesimistas.

El área que aborda la optimización de decisiones y simulación se conoce como **investigación de operaciones** o gestión de operaciones. Esta área, sin embargo, se ha ido fusionando con la analítica de negocios, principalmente por la popularidad que ha ganado esta última, pero también porque la modelación predictiva está tomando un rol protagónico en el proceso de toma de decisiones con herramientas cuantitativas.

La optimización ha sido mencionada varias veces a lo largo del libro: el proceso de minimizar el error en una red neuronal o regresión lineal es de optimización, al igual que la búsqueda de parámetros en SVM o la construcción de segmentos homogéneos en K-medias. Sin embargo, el foco de este capítulo está en cómo plantear problemas de negocios de forma cuantitativa como modelos de optimización.

Es importante distinguir dos etapas fundamentales: la **formulación** del problema, donde se plasma una situación a optimizar como una serie de ecuaciones, y la **optimización** propiamente tal, que es la etapa donde se resuelve el problema y se encuentra la solución óptima.

Empresas y organizaciones en diferentes industrias utilizan a diario modelos de optimización para responder preguntas como las siguientes:

- ¿Cómo planificar el transporte de diferentes bienes producidos desde las bodegas propias a los destinatarios finales, con el fin de satisfacer la **demanda de productos**, cumpliendo con los horarios y descansos de los choferes?
- ¿Cómo decidir el número de agentes en un *contact center*? ¿Cómo asignarlos de forma óptima con el fin de balancear su carga y horarios de trabajo, cumpliendo con las metas de atención en base a el **número de llamadas proyectadas**?
- ¿Cómo ubicar recursos de bomberos (cuarteles y carros) de tal forma de responder a siniestros en el menor tiempo posible en base a **proyecciones zonales de incendios y rescates**? (Pérez *et al.*, 2016).

Se destacan en negrita diferentes tareas de modelación predictiva que actúan como entrada para los modelos de optimización, y de esta forma se combinan ambos mundos con el fin de mejorar la toma de decisiones. Estas estimaciones de demanda y proyecciones se suelen realizar con modelos de series de tiempo (ver capítulo anterior), pero hay estrategias más sofisticadas que usan modelos avanzados de aprendizaje profundo (interpretación de imágenes o texto) como *input* para la optimización de decisiones en negocios.

La gestión de operaciones tradicional (el área encargada de modelar y optimizar las decisiones de negocios) dejó de ser el área cuantitativa *sexy* dentro de la empresa, siendo desplazada por la analítica. Entonces, ¿por qué un *data scientist* debería aprender optimización? Principalmente por dos razones:

- La modelación de problemas de negocios y la capacidad de abstracción que entrega aprender estas técnicas son fundamentales. Entender las decisiones es clave para definir y ejecutar todo el proceso de extracción de conocimiento en bases de datos. Si la analítica se realiza de forma aislada a las decisiones que se van a tomar, el proyecto está condenado al fracaso. La mejor manera de entender el proceso de toma de decisiones es ser capaz de modelarlo con abstracción matemática.
- La optimización es el pilar de los métodos de inteligencia artificial. Comprender lo que está detrás de los modelos predictivos permite entender por qué a veces estos son exitosos y otras veces fallan. Entender optimización y los fundamentos de los modelos evita que nos volvamos un *data hacker*, es decir, alguien que cree ser un *data scientist* pero solo es capaz de aplicar código sin entender la matemática que hay detrás.

### *10.1.1 Programación lineal*

A modo de ejemplo de formulación (muy simplificada), planteamos el siguiente problema de transporte logístico, en donde una empresa produce bienes en $m$ fábricas diferentes y debe satisfacer la demanda de $n$ diferentes locales de venta al detalle. Etiquetamos como $i = 1...m$ las diferentes fábricas y como $j = 1...n$ los diferentes detallistas.

Un modelo de programación lineal se caracteriza por un conjunto de decisiones a optimizar (maximizar o minimizar) en base a un objetivo y sujeto a una serie de restricciones. Para este problema en particular, tenemos:

- **El objetivo:** minimizar los costos de envío de productos. Se asume un costo lineal $c_{ij}$ por enviar una unidad de producto producida en la fábrica $i$ al detallista $j$.
- **La decisión:** cuántas unidades de producto enviar de la fábrica $i$ al detallista $j$. Nos referimos a esta decisión como $X_{ij}$.
- **Los datos/parámetros:** nos referimos con parámetros a la información de entrada del modelo. En este caso, sabemos que existe una capacidad de producción máxima $O_i$ de la fábrica $i$. Además, asumimos conocida una demanda futura $D_j$ de cada tienda al detalle, la cual puede ser estimada con modelos predictivos.
- **Las restricciones:** no se puede producir en una fábrica más que su capacidad máxima; además, debemos satisfacer la demanda de cada tienda al detalle.

La formulación del problema anterior queda de la siguiente forma:

$$min \sum_{j=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

sujeto a

Ecuación 10.1

$$\sum_{j=1}^{n} x_{ij} \leq O_i \ \forall_i, \sum_{i=1}^{m} x_{ij} \leq D_j \ \forall j \text{, y } x_{ij} \geq 0 \ \forall i, j$$

Es importante notar que el problema anterior solo será **factible** si la oferta total es mayor que la demanda total, es decir, $\sum_{j=1}^{n} D_j \leq \sum_{i=1}^{m} O_i$ En caso de que el modelo sea factible, se formará un espacio factible con todas las posibles soluciones para este problema.

La estrategia tradicional para resolver una formulación de programación lineal es utilizar el método Simplex. Este se basa en comparar los vértices del espacio factible que resulta al evaluar todas las restricciones y la función objetivo.

Si queremos encontrar el óptimo dentro de todos los puntos factibles, Simplex nos dice que debemos deslizar la función objetivo en la dirección de mayor utilidad. Justo en el borde donde esta función se sale del poliedro es donde los mejores puntos se encuentran. Lo genial de Simplex es que uno de estos puntos óptimos de borde siempre se encontrará en una esquina del poliedro. Gracias a esto, el problema se reduce a evaluar esquinas con una cierta lógica.

### *10.1.2 Programación entera, mixta y no lineal*

Muchas veces las decisiones son enteras y no numéricas. Por ejemplo, suena poco razonable que el costo de producción y transporte del problema anterior sea lineal en términos de la cantidad despachada. Hace sentido, por ejemplo, considerar un costo variable de producción que puede ser lineal y un costo fijo de despacho si y solo si se envía de la fábrica *i* a la tienda *j*.

Para solucionar este problema, se introducen variables de decisión *switch:* variables binarias que se activan solo en caso de que se decida enviar productos de *i* a *j*. Modelos que contienen solo variables binarias se conocen como de **programación entera**, mientras que aquellos que combinan tanto variables numéricas como de *switch* se conocen como de **programación mixta**.

Formalicemos el ejemplo anterior.

- **El objetivo:** minimizar los costos de envío y producción. Se asume un **costo variable** lineal $CV_{ij}$ por unidad de producción y

un **costo fijo** $cf_{ij}$ por enviar una unidad de producto producida en la fábrica i al detallista j.

- **Las decisiones:** la variable $X_{ij}$ representa las unidades de producto a enviar de la fábrica i al detallista j, mientras que la variable binaria $y_{ij}$ indica si se despacha o no de i a j.
- **Los datos/parámetros:** tal como en el ejemplo anterior, tenemos una capacidad de producción máxima $O_i$ de la fábrica *i*. Además, asumimos conocida una demanda futura $D_j$ de cada tienda al detalle, la cual puede ser estimada con modelos predictivos.
- **Las restricciones:** no se puede producir en una fábrica más que su capacidad máxima; por su parte, debemos satisfacer la demanda de cada tienda al detalle. Además, debemos imponer una condición que relacione ambas variables: si despachamos de *i* a *j*, es decir, $x_{ij} > 0$, entonces activa el *switch* ($y_{ij} = 1$). Este tipo de restricciones se conocen como de «gran M» *(big M)* ya que usan un valor muy grande llamado *M*. Este valor debe ser lo suficientemente grande para que nunca se despache más que *M*.

La formulación del problema anterior queda de la siguiente forma:

$$min \sum_{j=1}^{n} \sum_{j=1}^{n} (cv_{ij} x_{ij} + cf_{ij} y_{ij})$$

sujeto a

Ecuación 10.2

$$\sum_{j=1}^{n} x_{ij} \leq O_i \; \forall_i, \; \sum_{i=1}^{m} x_{ij} \leq D_j \; \forall j, \; x_{ij} \leq M y_{ij} \; \forall i,j, \text{ y } x_{ij} \geq 0, y_{ij} = \{0,1\} \; \forall i,j$$

Para finalizar, es importante indicar que hay modelos de **programación no lineal**, donde las variables de decisión se multiplican o dividen. Esto no está permitido en optimización lineal. Bajo ciertas condiciones, sin embargo, puede ralentizar la optimización.

Tomemos, por ejemplo, el problema de optimización de la regresión lineal llamado mínimos cuadrados ordinarios (MCO, ver capítulo anterior). En este problema, las variables de decisión son los coeficientes beta. Lo que hace MCO es calcular las diferencias (al cuadrado) entre la predicción y el valor real. Este proceso eleva las variables al cuadrado, lo que resulta en un caso particular de programación no lineal (programación cuadrática). Este problema es relativamente simple de optimizar, llegando a una ecuación exacta para los valores óptimos de beta.

Un caso diferente es una red neuronal, que multiplica los pesos *w* de la red de una forma mucho más compleja, lo que resulta en una programación no lineal más costosa de optimizar (ver capítulo 8). Este problema lo soluciona SVM al ser un caso simplificado de una red neuronal que devuelve el problema a uno de optimización cuadrática.

¿Por qué este hecho no resta popularidad a las redes neuronales? Debido a que el valor de la solución óptima no es tan alto en inteligencia artificial como en gestión de operaciones. Cuando queremos optimizar decisiones en la empresa, buscamos la mejor. Sin embargo, ya aprendimos el concepto de sobreajuste: si entrenamos demasiado una red, puede que la solución «aprenda demasiado» y no funcione adecuadamente en nuevos datos. Por esto, una red neuronal se entrena en un número determinado de iteraciones, monitoreando el desempeño en el conjunto de validación. En otras palabras, nos olvidamos de que existe un óptimo para el problema de optimización que resuelve una red neuronal, conformándonos con buenas soluciones.

### *10.1.3 Modelación del riesgo: optimización estocástica y simulación*

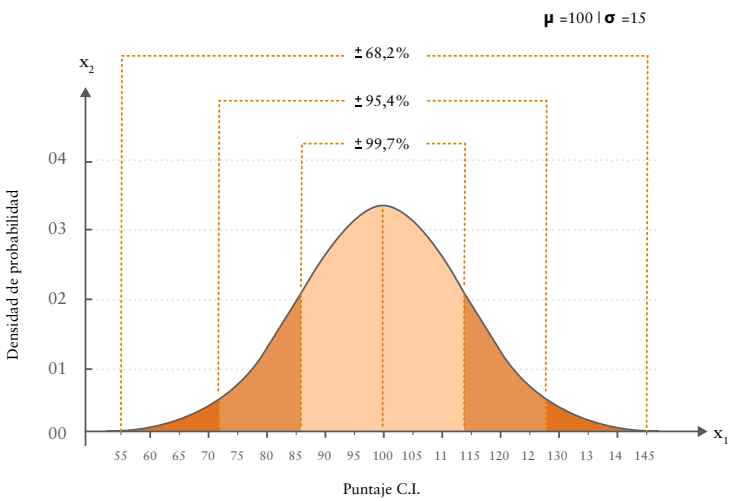
Hasta ahora hemos asumido que la optimización tiene *inputs* conocidos (los parámetros) que se comportan de manera estable. Esto difícilmente se sostiene en la realidad de las tareas que mencionamos al comienzo del capítulo: la demanda no se puede predecir

con exactitud, la llegada de los insumos no siempre es a tiempo y, así, una serie de imponderables que pueden afectar la optimización. En otras palabras, los datos de entrada contienen una variabilidad intrínseca.

Muchas veces esta incertidumbre se puede modelar. Un modelo predictivo no solo nos puede entregar la demanda pronosticada, sino también una **distribución.** A grandes rasgos, una distribución entrega la probabilidad de que ocurran diferentes eventos, en donde la suma de todas estas probabilidades es igual a 1.

La distribución más famosa es la normal, que es simétrica (por ejemplo, es igualmente probable subestimar la demanda que sobreestimarla) y con forma de campana (los escenarios más probables se encuentran cercanos al valor predicho). La distribución normal es muy importante, ya que en base a ella se derivan la mayoría de los *tests* estadísticos mencionados en el libro. Un ejemplo de una distribución normal es la distribución de coeficiente intelectual (CI) de las personas, la cual se muestra en la figura 10.1.

Figura 10.1: Distribución normal




Al entender ciertos elementos de la distribución de los parámetros con incertidumbre, podemos modelar de mejor manera el problema. A grandes rasgos, hay dos estrategias para abordar

este punto. La primera es la **optimización estocástica**, que sigue un camino más bien teórico para realizar formulaciones incluyendo elementos de la distribución. A modo de ejemplo, el enfoque min-max busca minimizar costos (u otro objetivo), asumiendo el peor escenario posible de las entradas con incertidumbre. Estos modelos se plantean de forma matemática y llegan a estrategias que pueden garantizar una solución óptima. Sin embargo, son conceptualmente complejos y suelen ser computacionalmente costosos en tiempos de ejecución. La solución óptima del problema min-max puede ser también demasiado conservadora.

Una segunda alternativa, más «empírica» que teórica, es el uso de simulación. La simulación de Monte Carlo permite tomar muestras de la distribución asumiendo ciertos parámetros como la media y la desviación estándar. Al simular valores de demanda, por ejemplo, se puede evaluar el óptimo del modelo bajo diferentes escenarios. Esto nos permite generar una distribución para nuestro objetivo, es decir, evaluar qué pasaría con nuestras decisiones bajo una cierta cantidad de escenarios generados (por ejemplo, 100 o 1000 escenarios). Esta distribución de los costos o las utilidades, dependiendo lo que estemos minimizando o maximizando, nos permite balancear riesgos y, quizás, tomar decisiones más conservadoras o agresivas de lo que resultaría si solo consideramos la demanda promedio.

## 10.2 Decisiones de umbral y medidas de *profit*

En el capítulo 6 discutimos diferentes medidas estadísticas para evaluar el desempeño de modelos de clasificación y compararlos entre ellos. Como medida principal sugerimos el área bajo la curva ROC (AUC), ya que permite elegir el mejor desempeño, independiente del umbral que utilicemos para tomar la decisión final del clasificador (a quién aceptar o rechazar en riesgo de crédito, o a quién ofrecerle una campaña de retención para evitar fuga).

Llegó el momento de hacernos cargo de la decisión más importante y final: el umbral o punto de corte. Todos los modelos predictivos son capaces de entregar una probabilidad de ocurrencia

de la clase en estudio (probabilidad de que un cliente sea un mal pagador, o que se fugue). Esta información es valiosa para tomar una decisión bien meditada.

### *10.2.1 Tabulación de posibles escenarios de corte y sus costos*

La primera solución es aplicar *descriptive analytics* y tabular los diferentes desenlaces que se obtienen al aplicar un punto de corte determinado. La solución estadística por defecto es considerar un umbral de 0.5. En riesgo de crédito, por ejemplo, esto quiere decir que todos los clientes con una probabilidad de incumplimiento crediticio mayor a 0.5 son rechazados, siendo aceptados aquellos bajo el umbral. Este corte de 0.5 es el de **máxima verosimilitud** en regresión logística, es decir, el más compatible con los datos de entrenamiento. Sin embargo, esto no garantiza que sea el que tenga el mayor acierto ni el que traiga más beneficios en el conjunto de predicción.

Antes de tabular los resultados, es importante tener una noción de los costos (o mejor aún, la utilidad neta) que resultan de aplicar el modelo. De esta forma podemos visualizar el efecto que tiene alejarnos de los puntos que maximizan el acierto global.

En riesgo de crédito, por ejemplo, tenemos dos costos importantes: el costo de clasificar un potencial buen cliente como mal pagador (y rechazarlo), y el costo de aceptar un cliente que en realidad resulta ser un mal pagador. El primer costo se asocia a los intereses que deja de percibir la entidad financiera por concepto del crédito no otorgado. Esto se conoce como un **costo de oportunidad**. Formalmente, se estima como el ROI (retorno de la inversión esperado o *return on investment*) del crédito multiplicado por el monto otorgado (Verbeke *et al.*, 2017).

Por otro lado, el segundo costo es equivalente a la pérdida monetaria de un crédito otorgado a un *defaulter*. Es importante entender que este valor no es exactamente el monto otorgado ya que hay importantes esfuerzos de cobranza de deudas. De acuerdo con la regulación vigente en la mayoría de los países, este costo corresponde a la LGD (pérdida por incumplimiento o *loss given*

*default*) multiplicada por la EAD (exposición en el momento del incumplimiento o *exposure at default*) (Baesens *et al.*, 2016).

La LGD se refiere a la fracción de la deuda que se pierde efectivamente después de las tareas de cobranza. Por ejemplo, si un cliente tiene una deuda pendiente de $100 y el banco logra recuperar $10 (incluyendo costos de cobranza), entonces la LGD es de un 90%. En créditos hipotecarios, por ejemplo, la LGD puede ser baja, ya que implica la venta de las garantías o colaterales por parte del banco. La EAD, en cambio, es la deuda pendiente del cliente al momento de darse el incumplimiento.

Algunos de los costos se pueden obtener a nivel de cliente al momento de otorgar (o no) el crédito, como el monto del crédito y sus intereses. Sin embargo, la deuda al momento de incumplimiento y la LGD son valores que solo se pueden obtener un tiempo después de la otorgación del crédito. Por esto, una solución es considerar valores promedios.

El uso de promedios en vez de costos por cliente puede ser más o menos realista dependiendo de la homogeneidad de los créditos otorgados que se consideran en el modelo. Si los datos son muy heterogéneos, una alternativa es segmentar a los clientes y usar valores promedio por segmento para estos *inputs* (Maldonado *et al.*, 2021).

Procedamos a tabular los diferentes escenarios de corte para nuestro ejemplo de riesgo de crédito reportado en el capítulo 6. Utilizando un modelo de regresión logística para calcular la probabilidad de incumplimiento, calcularemos el desenlace de la otorgación de créditos para un valor creciente del umbral. Partiremos en 0 (rechazar a todos) y llegaremos hasta 1 (aceptar a todos) en intervalos de 0.05. Simplificando aún más el problema, vamos a considerar una relación de costos de 1 es a 2, es decir, el costo de aceptar un cliente mal pagador es dos veces más caro que el costo de rechazar un cliente potencialmente «bueno». Es realista asumir que el riesgo de los *defaulters* es más alto que el costo de oportunidad de dar crédito.

Finalmente, podemos considerar un enfoque de tres zonas utilizando dos puntos de corte. Podemos llamarle un enfoque de semáforo: una zona verde de aceptación inmediata bajo el primer

umbral, una zona roja sobre el segundo umbral de rechazo automático, y una zona amarilla entre ambos umbrales. En esta última zona, la decisión final de otorgamiento puede pasar a un comité de expertos, o bien solicitar información adicional al postulante a crédito para evaluar sus méritos. Otras estrategias que utilizan las entidades financieras para controlar sus riesgos son otorgar créditos más pequeños o solicitar garantías a los postulantes.

Es importante entender las limitaciones del modelo. Cuando la probabilidad de incumplimiento que este entrega es, por ejemplo, 0.55, el modelo está diciendo que no sabe qué hacer. Al asumir un punto de corte de 0.5, «obligas» al modelo a rechazar a este cliente. Dependiendo de los niveles de riesgo de la entidad puede ser la decisión correcta, sin embargo, hay más opciones que solo aceptar o rechazar automáticamente. Esto es extrapolable al resto de las aplicaciones.

Cabe mencionar que formalizamos este concepto de «semáforo» en riesgo de crédito en base a la teoría de *rough sets* en el *paper* Maldonado *et al.* (2020). Este trabajo se basa en la teoría de *three-way decisions* del mundo de *rough sets,* que es popular en el ámbito académico y cuadra muy bien con la idea de dos umbrales. Incorpora además los costos de solicitar más información adicional a aquellos clientes que caigan en la zona amarilla con el fin de definir umbrales óptimos. No se menciona en detalle esta estrategia por ser una teoría matemáticamente sofisticada y que pertenece a un mundo que rara vez escapa de la academia.

### *10.2.2 Medidas de profit para fuga de clientes*

La búsqueda del umbral óptimo en diferentes aplicaciones de negocio llevó a formalizar esta estrategia en lo que se conoce como medidas de utilidad o *profit measures*. El trabajo de Verbeke *et al.* 2012 presentó el concepto de MP *(maximum profit)* para fuga de clientes, entendiendo que la mejor forma de abordar este problema es enfocarnos en el objetivo último de la tarea: la campaña de retención.

Siguiendo la fórmula de Neslin descrita en la Aplicación 3 (ecuación App3.2) para la utilidad **total** de campaña de retención, Verbeke *et al.* (2017) propusieron la siguiente ecuación para obtener la utilidad **promedio del clasificador** $P_c$ por cliente en función del punto de corte *t*. Esta función se basa en considerar cuatro escenarios clave:

- El modelo predice que a un cliente como «no fuga» y el cliente no recibe la campaña de retención. El costo o beneficio para este cliente es cero ya que no hay un flujo monetario, independiente si se fuga o no lo hace. En otras palabras, se considera como situación base no realizar una campaña y se evalúa cuánto mejorarían, o empeorarían, las utilidades al hacerla.
- Si el cliente es etiquetado **incorrectamente** como potencial fuga, «desperdiciamos» en él la campaña de retención. Aquí consideramos dos costos: el costo de contactar el cliente *f* (que suele considerarse muy bajo) y el costo del incentivo para retenerlo *d*. El costo de este escenario es entonces - *d* - *f*.
- Si el cliente es etiquetado **correctamente** como potencial fuga, puede ser que decida **quedarse** y aceptar la campaña de retención. Se asume una probabilidad γ de que el cliente tome esta decisión. Aquí se logra ganar el *jackpot* de la campaña, que es el LTV o valor presente de los beneficios que trae el cliente a futuro. A esto, eso sí, hay que restarle los costos de campaña: *LTV* - *d* - *f*.
- Si el cliente es etiquetado **correctamente** como potencial fuga, puede ser que decida **irse** pese a recibir la campaña de retención. En este caso solo se incurre en el costo de contacto *f* ya que el incentivo es rechazado. Se asume una probabilidad *1* -γ de que el cliente tome esta decisión.

El objetivo es entonces maximizar la utilidad, considerando el umbral *t* como variable de decisión y el resto de la información como parámetro. Podemos entonces escribir esto como un problema de optimización:

Ecuación 10.3

$$MPC = max_t P_c(t; \gamma, LTV, \delta, \Phi)$$

Retomando el tema del manejo de la incertidumbre, un parámetro clave de este modelo es γ, la probabilidad de que un cliente con intenciones de fugarse logre ser retenido por el esfuerzo de marketing. Podemos aplicar el concepto de optimización estocástica y ajustar la distribución de γ en base a datos históricos y supuestos, y utilizar como medida para elegir el umbral el **valor esperado** del *profit:*

Ecuación 10.4

$$EMPC = \int_{\gamma} P_c(t^*(\gamma); \gamma, LTV, \delta, \Phi) \cdot h(\gamma) d_\gamma$$

La integral en la ecuación 10.4 aparece ya que estima la esperanza de la distribución de γ. Esto se interpreta como la evaluación de la medida de *profit* frente a todos los posibles escenarios de γ. Una integral es una suma de escenarios en un espacio continuo, donde existen infinitas opciones, pero con una probabilidad ínfima de ocurrencia. El trabajo que propone esta medida sugiere el uso de una función de probabilidades Beta para *y*, que es una distribución continua (tal y como la Normal) pero mucho más flexible.

La investigación en medidas de *profit* ha ido más allá de la elección del umbral y ha incorporado costos y beneficios de clasificación dentro de los mismos modelos predictivos. El objetivo es construir clasificadores sensibles a los costos, que busquen maximizar un criterio de negocio más que uno estadístico. El primer acercamiento en esta dirección lo propusimos en SVM, utilizando medidas de *profit* para guiar la elección de parámetros y la selección de atributos (Maldonado *et al.*, 2015). Después han salido varias variantes de modelos predictivos que aprenden maximizando *profit,* adaptando técnicas como regresión logística, árboles de decisión y SVM (Verbeke *et al.*, 2017). Finalmente, hace poco propusimos un modelo de optimización robusta que maximiza el *profit* controlando el riesgo que presenta la distribución de los datos de entrada (Maldonado, López y Vairetti, 2020).

## 10.3 Aprendizaje causal

Otro tema que está en boga en *analytics* es el aprendizaje causal. Siguiendo el espíritu de este capítulo, la predicción en negocios no es un objetivo en sí mismo, sino que un medio para tomar mejores decisiones. Sin embargo, una buena predicción no garantiza una buena rentabilidad de la acción.

Supongamos que tenemos un modelo muy preciso de predicción de fuga y decidimos abordar, con una campaña de retención, a aquellos con una mayor probabilidad de dejar la compañía. Sin embargo, la gran mayoría de los clientes que reciben la campaña terminan fugándose pese a la campaña. ¿Qué pudo haber fallado?

Podemos, por un lado, culpar a la campaña, que no fue lo suficientemente convincente. Nuestro modelo, al final del día, funcionó muy bien, ¿no? Bueno. Podríamos haberlo hecho mejor como *data scientist*s si nos hubiésemos hecho la pregunta correcta: ¿Podemos identificar a los clientes con una mayor probabilidad de ser **convencidos** por la campaña de retención?

Este es el concepto principal en aprendizaje causal, y en particular, en el uso de medidas de *uplift:* predecir **persuasión**. Desde un punto de vista técnico, estas estrategias buscan predecir el efecto de un **tratamiento,** identificando aquellos individuos con mayor diferencia de respuesta si son tratados en comparación a si no lo son.

Llamaremos a quienes recibieron la campaña como grupo de tratamiento, y a quienes no la recibieron como grupo de control. Este concepto se conoce también como A/B *testing*. La estrategia de *uplift,* sin embargo, tiene una dificultad evidente: ¿cómo podemos saber realmente quién es convencible si solo observamos clientes fugados y no fugados dentro de cada grupo (tratamiento y control)? Esto es un fenómeno contrafactual.

Podemos reconocer los siguientes cuatro tipos de clientes, los cuales se observan en la figura 10.2:

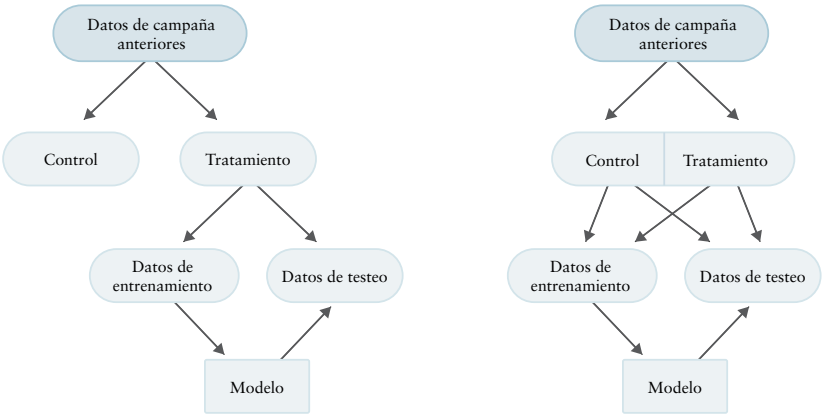
Figura 10.2: Tipos de clientes de acuerdo con el modelo de *uplift*

| | | Responde si NO recibe campaña | |
|---|---|---|---|
| | | Sí (NF) | No (F) |
| Responde si recibe campaña | Sí (NF) | Seguros | Influenciables |
| | No (F) | No molestar | Causas perdidas |

- **Seguros** *(sure things):* responden positivamente (no se fugan), independiente si reciben el tratamiento o no (en nuestro caso, la campaña de retención). El incentivo no genera retornos adicionales en ellos y tienden a hacernos creer que la campaña ha sido exitosa, cuando en realidad puede no serlo. De todos modos, hay un beneficio indirecto al incrementar la satisfacción y lealtad de clientes ya comprometidos con la empresa (por ejemplo, pueden volverse **promotores** – ver Aplicación 4), pero se excluye del análisis por simplicidad.
- **Influenciables** *(persuadables):* aquellos clientes que responden positivamente solo si reciben la campaña; en caso contrario, se fugan. Estos clientes son el objetivo del modelo y deben ser identificados para recibir la campaña.
- **Causas perdidas** *(lost causes):* estos clientes se fugan independiente si reciben la campaña de retención o no. No generan beneficios ni costos, salvo el de contactar al cliente. Dado que los seguros terminan aceptando la campaña, se incurre en un costo mayor en estos últimos, ya que las causas perdidas tienden a rechazar el incentivo.
- **Los «no-molestar»** *(do-not-disturbs):* también conocidos como *sleeping dogs*, son aquellos que no responden (se fugan) a causa del tratamiento, es decir, la campaña les recuerda sus intenciones de fugarse. Este es el mayor costo de la campaña, ya que perdemos el LTV del cliente y, por ende, debemos evitar ofrecer la campaña a ellos a toda costa.

Es imposible observar todos los comportamientos de los clientes y, por supuesto, meternos en sus mentes y averiguar si fue la campaña la que logró convencerlos o nunca habían pensado en irse. A grandes rasgos, lo que buscan los modelos de *uplift* es aplicar dos modelos de fuga: uno construido en base al conjunto de tratamiento y otro al grupo de control. Esto se muestra en la figura 10.3 (lado derecho). Notar que el enfoque tradicional (figura 10.3, lado izquierdo) es realizar la construcción de modelos de clasificación solo con quienes recibieron la campaña de retención, o simplemente ignorar la información de campañas previas. El foco está, entonces, en predecir quién está en riesgo de dejar la compañía, independiente de si recibirá o no un incentivo para quedarse en ella.

Figura 10.3: Esquema de aprendizaje con datos de campañas de marketing (tradicional vs. *uplift*)




ESQUEMA TRADICIONAL ESQUEMA UPLIFT

Aplicamos ambos modelos a los clientes objetivo candidatos a recibir la campaña y calculamos:

Ecuación 10.5

$$uplift = P[respuesta \mid tratamiento] - P[respuesta \mid control]$$

Un cliente con un *uplift* alto es atractivo para la compañía ya que es un **influenciable.** Por ejemplo, si se estima que su probabilidad de fuga es 0.6 si no recibe campaña de retención (*1 - P* [*respuesta* | *control*])y 0.3 si la recibe (*1 - P* [*respuesta* | *tratamiento*]), entonces la campaña tuvo un gran impacto positivo en él (*uplift* = 0.3).

Si el *uplift* es cercano a cero, entonces es un cliente seguro o causa perdida, dependiendo de la probabilidad de fuga. Por ejemplo, si la probabilidad de fuga baja de un 0.95 sin campaña a un 0.90 con ella (*uplift* = 0.05), el cliente debe considerarse una **causa perdida.** Lo paradójico es que cualquier modelo de fuga tradicional diría que debemos gastar recursos en él. Por otro lado, si la probabilidad de fuga baja de un 0.1 sin campaña a un 0.05 con ella (*uplift* = 0.05), el cliente es **seguro** y no amerita enfocar el esfuerzo de marketing en él.

Finalmente, el *uplift* puede ser negativo, es decir, podemos «dispararnos en el pie» con la campaña y convencer a un cliente de que se vaya por el solo hecho de recibir la campaña. Si la probabilidad de fuga sube de un 0.4 sin campaña a un 0.7 con ella (*uplift* = -0.3),el cliente es un «**no-molestar**» y debemos asegurarnos de que no reciba la campaña, ya que podemos «despertarlo».

Puede sonar irrisorio que un cliente se fugue por el mero hecho de ofrecerle algo, pero ocurre. Puedes recordarle, por ejemplo, que tiene un servicio que ya no usa o con alguna tecnología obsoleta, como telefonía fija. Otro ejemplo es la industria de las inversiones y fondos mutuos. Imagina recordarle a un cliente que sus inversiones se han devaluado por alguna crisis, pero, como es el mensaje tradicional en este tipo de industrias, las fluctuaciones de mercado son algo normal y lo importante es pensar en el largo plazo. El cliente puede no entender el mensaje, asustarse y retirar los fondos (Maldonado *et al.*, 2021).

Se han reportado varias aplicaciones del modelo de *uplift* además de fuga de clientes, las cuales se describen brevemente a continuación (Siegel, 2013; Verbeke *et al.*, 2017):

- **Modelos de respuesta (incidencia de compra):** este es, quizás, el enfoque más popular de esta técnica. Por lo mismo, se refiere a ella como modelos de respuesta: buscamos predecir qué clientes

responderán favorablemente a una oferta y comprarán un producto o servicio. Esto es válido para nuevos clientes (adquisición de clientes) y existentes (ventas cruzadas).

- **Modelos de cobranza de deuda:** ¿a qué deudor deberíamos contactar con el fin de maximizar los retornos de los esfuerzos de cobranza?
- **Contacto multicanal:** ¿por qué medio deberíamos contactar a un cliente (carta certificada, *email*, llamada telefónica, mensaje de texto, etcétera) para que responda positivamente? Esto aplica prácticamente a cualquier tarea de un *contact center,* incluyendo prevención de fuga, incidencia de compras y cobranza.
- **Ensayos clínicos y medicina personalizada:** el concepto de A/B *testing* y usos de tratamiento y control son comunes en medicina desde hace ya bastante tiempo. Buscan evaluar el efecto neto de un nuevo medicamento o terapia en la salud del paciente. Esto puede depender de los atributos del paciente y derivar en medicina personalizada.
- **Campañas políticas:** la modelación predictiva en política se ha vuelto un tema popular en el área (pero también de preocupación; ya volveremos a este tema en el siguiente capítulo). El foco es diseñar propaganda personalizada enfocada en los ciudadanos correctos, es decir, los influenciables que no votarían por un determinado candidato si no reciben esta propaganda, pero sí lo harían en caso de recibirla (los votantes indecisos o *swing voters*).

Existen diferentes tipos de modelos de *uplift,* adaptaciones de modelos de clasificación presentados en los capítulos 6 y 7. Estos enfoques los podemos clasificar de la siguiente manera:

- **Enfoques de dos modelos:** podemos construir dos modelos independientes en base a los conjuntos de tratamiento y control. Los clientes cuyo comportamiento buscamos predecir se evalúan en ambos modelos, se calcula el *uplift* como la resta de la probabilidad de respuesta entre el modelo de tratamiento y control (ecuación 10.5) y se «rankea» a los clientes de mayor

a menor *uplift*. Este enfoque es independiente de la elección del clasificador.

- **Enfoques transparentes de un modelo:**
  - El modelo de Lo es una extensión de la regresión logística, en donde se predice si el cliente responde o no (enfoque tradicional). Sin embargo, en la matriz *X* se incluyen dos conjuntos de variables: el *set* original y las mismas variables multiplicadas por *t*, una variable *dummy* que indica si el cliente pertenecía al conjunto de tratamiento o de control. De esta forma se puede modelar el efecto de interacción asociado a la respuesta. Finalmente, se propone una eliminación *stepwise* de variables (ver capítulo 6) para reducir el conjunto de variables en el modelo, considerando que fue duplicado (Olaya *et al.*, 2020).
  - El modelo de Kane aborda el problema como una tarea multiclase (logit multinomial, ver capítulo 7), donde se predicen cuatro clases: *control responders*, *control non-responders*, *treated responders*, *treated non-responders* (Olaya *et al.*, 2020).
  - Existen además variantes de modelos de árboles de decisión, tales como el *significance-based uplift tree* (SBUT) o el *divergence-based uplift tree* (DBUT). Estos modelos siguen la lógica de los árboles originales, pero definen medidas de ganancia para elegir los mejores cortes usando variantes de la medida de *uplift* (Olaya *et al.*, 2020).

- **Enfoques de caja negra de un modelo:** los más populares son los métodos multiclasificadores o *ensembles*, como extensiones de *random forest* y *XGboost* (ver capítulo 7). Estos modelos, en general, siguen el principio detrás de los modelos de *ensemble*, pero basándose en adaptaciones de *uplift trees*. Estrategias populares son el uplift *random forest* o el *Causal Conditional Inference Forest* (CCIF).

Un aspecto clave de los modelos de *uplift* es que asumen que la asignación de personas a los grupos de tratamiento y control es al azar. Esto se conoce como un experimento «randomizado» o aleatorio. Si bien esta es la norma en ensayos clínicos, rara vez observaremos en negocios una división natural de tratamiento y control. Esto ocurriría, por ejemplo, si una empresa decide enviar incentivos de retención u ofertas al azar para evaluar el comportamiento de la campaña. Lamentablemente, muy pocas empresas se animarían a hacerlo de esta forma.

La buena noticia es que existen soluciones del mundo de la estadística para manejar el **sesgo de selección** que ocurre cuando los experimentos no son al azar. Una solución popular que busca alivianar este problema es el **pareamiento por puntaje de propensión** o *Propensity Score Matching* (PSM). El objetivo de este método es emparejar cada individuo tratado (se asume que son los minoritarios) a uno o más del grupo de control utilizando, por ejemplo, alguna técnica de vecinos más cercanos (ver capítulo 7). Los clientes no pareados del grupo de control son descartados del conjunto de entrenamiento. De esta forma se equilibran los grupos de tratamiento y control en términos de los valores promedio de las variables que la componen, sin sacrificar una gran cantidad de datos. Existen también otras estrategias que evitan eliminar datos, como el método *synthetic control,* que introduce ponderadores de tal forma que el grupo de control se asemeje en distribución lo más posible al de tratamiento (Olaya *et al.*, 2020).

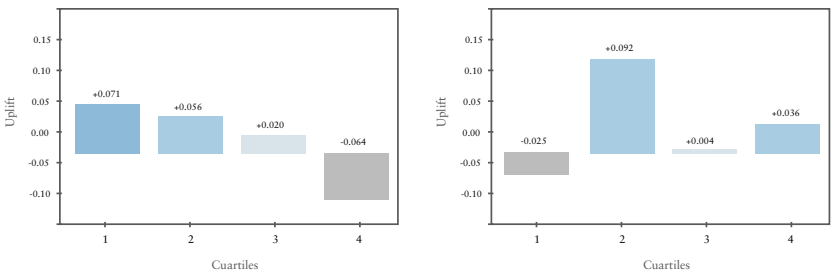
Finalmente, es importante mencionar cómo se evalúan los modelos de *uplift*. Una forma es gráfica, adaptando las curvas *lift*. Estas son esencialmente gráficos de barra en donde se divide la salida probabilística del modelo en deciles u otro tipo de percentiles, definiendo segmentos con mayor o menor probabilidad de fuga. Luego se grafica la tasa de fuga real (o de respuesta, en el caso de incidencia de compra) para cada decil. Si los primeros deciles tienen una tasa de fuga efectiva mucho mayor que los siguientes, entonces el modelo está discriminando correctamente.

En vez de representar la tasa de fuga por decil, podemos desagregar las columnas por tratamiento y control. El *uplift* se representa como la diferencia entre las barras de tratamiento y control. Esta diferencia se puede graficar por sí sola, en un gráfico de *uplift* por decil.

En la figura 10.4 se contrasta el desempeño de un modelo calibrado con un enfoque *uplift* (figura 10.4 (a)) y uno calibrado de forma tradicional para fuga de clientes (figura 10.4 (b)) agrupando por cuartiles de clientes. Dado que en el primer enfoque se ordena por *uplift* en la construcción del modelo, vemos que este decrece también en modalidad predictiva (figura 10.4 (a)). Esto permite elegir el primer cuartil para la campaña de retención y asegurar que se esté enfocando principalmente en los influenciables.

En el enfoque tradicional, en cambio, encontramos muchas causas perdidas en el primer cuartil al ordenar los clientes de acuerdo con su probabilidad de fuga de mayor a menor. Esto refleja que la fuga es alta, tanto en los grupos de tratamiento como de control, y un *uplift* cercano a cero (figura 10.4 (b)). El cuartil siguiente logra identificar correctamente influenciables, aunque que el comportamiento del *uplift* es más bien errático en general (Olaya *et al.*, 2020).

Figura 10.4: Curva *uplift* para dos modelos, uno de *uplift* (Modelo a) y otro predictivo tradicional (Random Forest, Modelo b)




(a) Uplift (MOAxgboost)

(b) Random forest

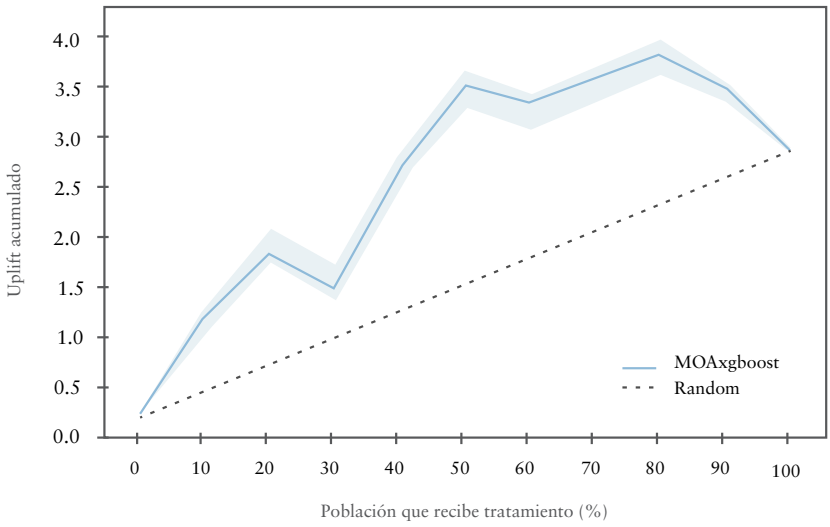
Alternativamente, el desempeño de un modelo de *uplift* puede visualizarse de forma acumulada, evaluando el *uplift* en el eje *Y* a medida que se incrementa el porcentaje de clientes a los cuales se le ofrece la campaña (de 0 a 100%). La curva que forman los valores del *uplift* acumulado se conoce como curva Qini. Hay una similitud entre

la curva Qini y la curva ROC, ya que todos los modelos que se comparen parten y terminan en los mismos puntos (0% de clientes lleva a un *uplift* de cero, mientras que si se contactan a todos los clientes llegaremos al mismo *uplift* independiente del clasificador). Un modelo al azar implica un crecimiento lineal en *uplift* desde el 0% al 100% de clientes contactados, lo que es similar a la diagonal de la curva ROC.

La figura 10.5 ilustra la curva Qini para los modelos destacados en las figuras anteriores, además del caso basal (lineal punteada, selección al azar). Si bien todos llegan al mismo *uplift* global de los datos (X%) al seleccionar todos los clientes, el *uplift* óptimo ocurre en los modelos 1 y 2 con un porcentaje mucho menor de clientes contactados, mostrando la utilidad de esta estrategia no solo para elegir el mejor modelo, sino también para el punto de corte óptimo o tamaño de la campaña (Olaya *et al.*, 2020).

Al igual que para la curva ROC, el área bajo la curva Qini permite resumir toda la información presentada en el gráfico y elegir el mejor modelo, independiente del umbral. Este valor se conoce como *medida Qini*. De todos modos, es una buena estrategia apoyarse en la curva Qini para la elección del umbral, que además depende de los presupuestos de campaña.

Figura 10.5: Curva Quini para dos modelos, uno al azar y otro de *uplift*



## Referencias

Baesens, B., Roesch, D. & Scheule, H. (2016). *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS*. New Jersey: John Wiley and Sons.

Maldonado, S., Flores, A., Verbraken, T., Baesens, B. & Weber, R. (2015). «Profit-based Feature Selection using Support Vector Machines - General Framework and an Application for Customer Churn Prediction». *Applied Soft Computing,* 35, 740-748.

Maldonado, S., López, J. & Vairetti, C. (2020). «Profit-based Churn Prediction based on Minimax Probability Machines». *European Journal of Operational Research,* 284(1), 273-284.

Maldonado S., Peters, G. & Weber, R. (2020). «Credit Scoring using Three-Way Decisions with Probabilistic Rough Sets». *Information Sciences,* 507, 700-714. *Special issue on Granular Computing, Shadowed Sets, and Three-Way Decisions*.

Maldonado, S., Domínguez, G., Olaya, D. & Verbeke, W. (2021). P*rofit-driven Churn Prediction for the Mutual Fund Industry: a Multisegment Approach*. *Omega* 100, ID 102380.

Olaya, D., Vásquez, J., Maldonado, S., Miranda, J. & Verbeke, W. (2020). «Uplift Modeling for Preventing Student Dropout in Higher Education». *Decision Support Systems,* 134, 113320.

Pérez, J., Maldonado, S. & Marianov, V. (2016). «A Reconfiguration of Fire Station and Fleet Locations for the Santiago Fire Department». *International Journal of Production Research,* 54(11), 3170-3186.

Siegel, E. (2013). *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die*. New Jersey: Wiley.

Verbeke, W., Baesens, B. & Bravo, C. (2017). *Profit Driven Business Analytics*. New Jersey: Wiley.

# Laboratorio 4: Predicción y decisión de umbral

En este último laboratorio, aplicaremos el mejor modelo encontrado en el Laboratorio 3 con los datos de crédito (SVM) sobre nuevos clientes cuyo comportamiento no conocemos. De esta forma emulamos lo que sería una eventual aplicación de un modelo predictivo.

Primero importamos las librerías a utilizar:

```
import pandas as pd import numpy as np
from sklearn.preprocessing import MinMaxScaler
import pandas.util.testing as tm from joblib import dump, load
```

El siguiente paso es montar Google Drive para poder enlazar nuestra base de datos. De esta forma podremos cargar nuestra base de datos creditSIM.xlsx (que debemos tener en una carpeta «Colab Notebooks» en nuestro Drive claramente identificada).

```
from google.colab import drive drive.mount('/content/gdrive')
```

Cargamos el archivo con la base de datos de testeo (segunda hoja del archivo Excel):

```
ruta_data_set = 'gdrive/My Drive/creditSIM.xlsx'
data_set = pd.read_excel(ruta_data_set, sheet_name = 'prediccion')
```

Cargamos ahora el archivo con el modelo entrenado:

```
modelo_entrenado = load('gdrive/My Drive/modelo_entrenado.joblib')
```

Nuestro objeto *data_set* tiene las variables independientes que nos ayudarán a predecir la variable objetivo (si cliente cae en *default* o no).

Antes de aplicar el modelo, debemos replicar el preprocesamiento a este nuevo *set* de datos.

## 1. Preprocesamiento de datos de predicción

Veamos qué contiene nuestra base de datos de predicción:

```
print(data_set)
```

```
        ID  AgnosDirec  AgnosEmpleo  ...      Ingreso2  Nacionalidad  NivelEdu
0   103001         9.0          5.0  ...  61359.560021             N    SupCom
1   103002         4.0          4.0  ...  21402.359671             N    SupInc
2   103003         8.0          4.0  ...  70155.668032             N       Med
3   103004         1.0          2.0  ...  31178.580505             N    SupCom
4   103005        11.0          7.0  ...  58437.844886             N       Bas
..     ...         ...          ...  ...           ...           ...       ...
995 103996         7.0          6.0  ...  46646.838382             N       Med
996 103997         4.0          5.0  ...  41824.930145             N       Bas
997 103998        13.0          3.0  ...  67196.496621             N    SupInc
998 103999         6.0          3.0  ...   8886.419312             N       Bas
999 104000         0.0          0.0  ...   6797.606907             N       Bas
```

```
data_set.describe(include='all')
```

Out[7]:

| | ID | AgnosDirec | AgnosEmpleo | DeudaExt | DeudaInt | Edad | Ingreso | Ingreso2 | Nacionalidad | NivelEdu |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1000.000000 | 999.000000 | 998.000000 | 999.000000 | 998.000000 | 1000.000000 | 999.000000 | 1000.000000 | 1000 | 1000 |
| unique | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1 | 5 |
| top | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | N | SupInc |
| freq | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1000 | 334 |
| mean | 103500.500000 | 7.355355 | 6.913828 | 3914.243122 | 2065.436561 | 36.789000 | 58707.868428 | 49285.833051 | NaN | NaN |
| std | 288.819436 | 7.325584 | 6.545925 | 5529.459045 | 4547.314237 | 9.808297 | 46424.137015 | 39255.939446 | NaN | NaN |
| min | 103001.000000 | 0.000000 | 0.000000 | 41.538809 | 5.553404 | 18.000000 | 6829.598681 | 0.000000 | NaN | NaN |
| 25% | 103250.750000 | 2.000000 | 2.000000 | 947.274416 | 381.562707 | 29.750000 | 28656.648227 | 23795.774298 | NaN | NaN |
| 50% | 103500.500000 | 5.000000 | 5.000000 | 2133.930304 | 900.497434 | 36.000000 | 46762.745843 | 39010.929379 | NaN | NaN |
| 75% | 103750.250000 | 10.000000 | 10.000000 | 4575.723755 | 2137.803214 | 44.000000 | 75077.549495 | 61567.786032 | NaN | NaN |
| max | 104000.000000 | 44.000000 | 37.000000 | 58258.459493 | 66931.246918 | 68.000000 | 509306.055393 | 395355.747185 | NaN | NaN |

La estadística descriptiva indica que tenemos un ingreso cero (inconsistencia) y algunos valores perdidos:

```
data_set.loc[data_set['Ingreso2'] < 1, ['Ingreso2']] = np.nan
```

Reemplazamos los valores perdidos por la media:

```
data_set['AgnosEmpleo'].fillna(data_set['AgnosEmpleo'].median(skipna=True),i
nplace=True)
data_set['AgnosDirec'].fillna
(data_set['AgnosDirec'].median(skipna=True),inplace=True)
data_set['Ingreso'].fillna
(data_set['Ingreso'].median(skipna=True),inplace=True)
data_set['Ingreso2'].fillna
(data_set['Ingreso2'].median(skipna=True),inplace=True)
data_set['DeudaInt'].fillna
(data_set['DeudaInt'].median(skipna=True),inplace=True)
data_set['DeudaExt'].fillna
```

```
(data_set['DeudaExt'].median(skipna=True),inplace=True)
```

Creamos la variable de endeudamiento (*ratio* deuda total sobre ingreso):

```
data_set['DeudaIng'] = (data_set['DeudaInt'] + data_set['DeudaExt'])/data_
set['Ingreso2']
```

Ahora que no tenemos valores perdidos ni inconsistencias en nuestra base de datos, transformamos las variables financieras a sus logaritmos (debemos hacer EXACTAMENTE lo mismo que en el modelo original).

```
data_set['Ln_Ingreso'] = np.log(data_set['Ingreso']+1)
data_set['Ln_Ingreso2'] = np.log(data_set['Ingreso2']+1)
data_set['Ln_DeudaExt'] = np.log(data_set['DeudaExt']+1)
data_set['Ln_DeudaInt'] = np.log(data_set['DeudaInt']+1)
del data_set['Ingreso']
del data_set['Ingreso2']
del data_set['DeudaExt']
del data_set['DeudaInt']
```

Creamos ahora un nuevo *dataframe*, X_pre, que tendrá las siete variables relevantes que incluía el modelo de SVM.

```
X_pre = pd.DataFrame(data_set[['AgnosDirec','AgnosEmpleo','Edad',
'DeudaIng','Ln_Ingreso2','Ln_DeudaExt','Ln_DeudaInt']])
X_pre.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   AgnosDirec   1000 non-null   float64
 1   AgnosEmpleo  1000 non-null   float64
 2   Edad         1000 non-null   int64
 3   DeudaIng     1000 non-null   float64
 4   Ln_Ingreso2  1000 non-null   float64
 5   Ln_DeudaExt  1000 non-null   float64
 6   Ln_DeudaInt  1000 non-null   float64
dtypes: float64(6), int64(1)
memory usage: 54.8 KB
```

Normalizamos la base de datos:

```
scaler = MinMaxScaler()
scaler_data = scaler.fit(X_pre)
```

```
X = pd.DataFrame(scaler_data.transform(X_pre), index= X_pre.index, columns= X_pre.columns)
```

## 2. Predicción

Con el modelo de entrenamiento y las variables independientes definidas y procesadas, podemos realizar la predicción de la variable objetivo, como se muestra a continuación:

```
prediccion = modelo_entrenado.predict(X)
print(prediccion)
```

```
[0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0
 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1
 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0
 1 0 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0
 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 0 1 0 0 0
 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0
 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 1
 0 0 1 0 1 1 0 1 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 1 0 1 1
 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1
 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 0 1
 1 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1 0 1
 0 0 1 1 0 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0
 1 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1
 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 1
 0 0 1 0 1 1 1 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0
 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1
 0 0 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0 0
 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 1
 0 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0
 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0
 0 0 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1
 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 0
 0 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0
 0 1 0 1 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0
 0]
```

Vemos los resultados de predicción si imprimimos el objeto «predicción», que toma el valor 1 cuando el cliente se predice como mal pagador («*default*» o «s») y 0 en caso contrario («no *default*» o «n»).

Para realizar esta predicción, el modelo utiliza un umbral de 0.5 para la probabilidad de incumplimiento. Esto quiere decir que

si p>0.5 «predicción» toma el valor de 1, mientras que cuando sea menor a este umbral toma el valor de 0.

¿Pero cuál es la mejor decisión de umbral? Contamos la cantidad de clientes predichos que caen en *default:*

```
prediccion.sum()/prediccion.size
```

```
0.383
```

Recordemos que la tasa de clientes malos es de un 20% en la base original. Nuestro modelo predice, sin embargo, más de un 38% de clientes malos pagadores. Esto se debe a la corrección que realiza SMOTE, que «hace creer» al modelo que existe la misma cantidad de clientes buenos y malos.

Podemos modificar este umbral según lo que necesitamos, definiendo, por ejemplo, el umbral de 0.6 para tener una política de otorgamiento menos estricta. Una política muy restrictiva puede hacernos perder participación de mercado.

```
prediccion_2 = (modelo_entrenado.predict_proba(X)[:,1] >= 0.6).astype(int)
print(prediccion_2)
```

```
[0 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0
 1 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0
 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0
 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0
 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 1 0
 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1
 0 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
 0 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 1 0 1
 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0
 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0
 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1 0
 0 0 1 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0
 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 1
 0 0 1 0 1 1 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0
 0 1 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 0
 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1
 0 0 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0
 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1
 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0
 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 0 0
 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0
 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1
 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0
 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0
 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0
 0]
```

Nuevamente contamos la cantidad de *defaulters* con este nuevo umbral:

```
prediccion_2.sum()/prediccion_2.size
```

```
0.338
```

Podemos ver que los *defaulters* disminuyen al aumentar el umbral (desde un 38 a un 33.8%). Es posible modificar este criterio según los costos y beneficios del modelo.

## 3. Exportar datos a Excel

Finalmente, armamos la base de datos con todas las variables originales y le agregamos la variable predicha para tener toda la información del cliente. Luego exportamos estos datos a Excel, para trabajar con ellos.

Primero tenemos que pasar las variables que transformamos a logaritmos a su escala original:

```
data_set['Ingreso'] = np.exp(data_set['Ln_Ingreso'])-1
data_set['Ingreso2'] = np.exp(data_set['Ln_Ingreso2'])-1
 data_set['DeudaExt'] = np.exp(data_set['Ln_DeudaExt'])-1
 data_set['DeudaInt'] = np.exp(data_set['Ln_DeudaInt'])-1
del data_set['Ln_Ingreso']
del data_set['Ln_Ingreso2']
del data_set['Ln_DeudaExt']
del data_set['Ln_DeudaInt']
```

Con ellas listas, agregamos la variable de predicción a nuestra base de datos:

```
data_set['VarObj'] = prediccion_2
```

Por último, exportamos la base de datos a Excel y la guardamos en nuestro *Drive:*

```
data_set.to_excel('creditSIM_prediccion.xlsx')
!cp creditSIM_prediccion.xlsx «gdrive/My Drive/»
```

# 11. Conclusión

Estamos llegando al final de este libro. Esperamos que, en este viaje por la analítica de negocios, hayas aprendido. Quizás llegaste con conocimientos matemáticos sólidos y pudiste comprender mucho mejor cómo aplicarlos en el mundo de los negocios, o bien traías cierta experiencia profesional de analista y pudiste ir más allá del mundo tradicional de reportes y planillas. De cualquier forma, ahora debes ser un gran conocedor de *analytics*, dotado con una caja de herramientas bien completa para enfrentar problemáticas de negocios.

En este libro cubrimos un amplio espectro de tópicos, pasando por estadística y visualización tradicional hasta modelación predictiva avanzada basada en las últimas técnicas de aprendizaje profundo. Nos sumergimos en el mundo del Big Data y vimos aspectos clave de la gestión de operaciones y toma de decisiones, como la optimización y la simulación. Finalmente, nos «ensuciamos las manos» con talleres en Python para aplicar nuestros conocimientos.

Nos gustaría usar este capítulo final para salir del mundo técnico y plasmar nuestros pensamientos sobre la práctica de la analítica de negocios e IA en el mundo real. Comenzaremos con algunas sugerencias para la implementación exitosa de modelos y su monitoreo a lo largo del tiempo. Luego, discutiremos una serie de mitos y leyendas que giran alrededor de la IA, aportando nuestra opinión al respecto. El libro finaliza con una recomendación de recursos complementarios que te permitirán profundizar en los diferentes tópicos abordados en el libro.

## 11.1 Implementación y monitoreo de modelos

Existen una serie de elementos clave para la implementación exitosa de modelos de analítica de negocios que rara vez se mencionan en libros técnicos de aprendizaje de máquinas, pues más bien surgen de la experiencia en proyectos. A continuación, presentamos una hoja de ruta informal con recomendaciones para maximizar las probabilidades de éxito con la analítica:

- **Compromiso de la alta gerencia:** el punto de partida del éxito en la analítica de negocios es un apoyo completo del *top management* a la toma de decisiones basada en datos. Muchas empresas tienen un enfoque de gestión centrado en la intuición, donde el manejo de datos se limita a mantener las bases de datos transaccionales actualizadas y a reportes simples. Las iniciativas de *business analytics* comienzan a aparecer de forma *bottom-up,* es decir, en ciertas jefaturas de departamento. El logro de estas iniciativas suele ser moderado. Migrar de este esquema a uno centrado en analítica requiere un cambio organizacional importante, pues debe incluir un área dedicada a la administración de datos con fines analíticos (Venkatesan *et al.*, 2015).
- **Conocer el problema y el contexto local:** la brecha entre quienes construyen los modelos y quienes toman las decisiones suele ser más grande de lo que parece. Esto es particularmente cierto con consultores externos, pero incluso dentro de la organización. Muchas veces se empuja la analítica sin claridad sobre la problemática que se quiere resolver, se construyen modelos predictivos que se utilizan con un propósito diferente al cual fueron diseñados, o simplemente modelos que pasan al olvido una vez que se construyen. Para reducir estas brechas es clave trabajar codo a codo con los tomadores de decisiones y validar con ellos los patrones encontrados, además de tener entrevistas con los encargados de aplicar los modelos y recolectar datos (los ejecutivos captadores de crédito en la aplicación de *credit scoring,* por ejemplo).

- **El valor del *descriptive analytics*:** es común que los científicos de datos busquen abordar la tarea predictiva con demasiada premura y «metan todo a la licuadora», como analogía a los modelos multivariados. Es importante tomarse el tiempo de visualizar los datos, encontrar patrones univariados y bivariados y, en base a este conocimiento, diseñar variables explicativas de forma inteligente. En ese sentido, muchos autores reconocen que el éxito de la analítica de negocios depende mucho más de las variables que se construyan en la etapa de transformación (capítulo 2) que de la elección del modelo predictivo (Baesens *et al.*, 2015). En otras palabras, *dashboarding* y estadística descriptiva no son fines en sí mismos, que se utilizan de forma alternativa a la modelación predictiva y prescriptiva, sino pasos clave en el proceso de extracción de conocimiento mediante ellos.
- **Foco en causalidad:** el comportamiento de los clientes que debemos predecir es aquel que trae un mayor impacto a la toma de decisiones. Como mencionamos en el capítulo 9, es mucho más atractivo predecir a qué clientes puedo convencer de que se queden en la empresa, por sobre predecir aquellos que se van a fugar. Es muy importante definir adecuadamente el objetivo de la tarea predictiva en base a las acciones que se buscan llevar a cabo, ya que existen varias alternativas de variables objetivo más allá de las evidentes.
- **Foco en los beneficios y costos que trae la implementación del modelo:** En la misma línea, las medidas estadísticas no nos permiten evaluar correctamente las acciones que se derivan de los modelos de analítica. Medidas como AUC son buenas para comparar entre modelos predictivos y elegir el de mejor desempeño, pero es clave hacer un análisis de costo-beneficio para tomar la mejor decisión de umbral una vez que hemos seleccionado la técnica de modelación. La tendencia del *prescriptive analytics* va incluso más allá, pues utiliza optimización y simulación para mejorar la toma de decisiones, siendo la modelación predictiva un *input* de estas estrategias y no un fin en sí mismo.

Es importante notar que un modelo predictivo puede tener un desempeño regular, incluso decepcionante, y, sin embargo, traer importantes beneficios a las empresas una vez que se aplican. Cuando se busca predecir el comportamiento de los clientes, es normal tener desempeños que son solo un poco mejores que lanzar una moneda al aire (por ejemplo, un AUC de 0.7). Esto es normal considerando el comportamiento muchas veces impredecible y errático de los clientes, imposible de capturar por completo con las variables explicativas que se disponen. Esto no implica que la modelación haya fracasado: este modelo puede alimentar una herramienta de toma de decisiones basada en optimización que logre mejorar por mucho las prácticas vigentes de una empresa basadas en intuición.

- **Postimplementación y monitoreo:** el mundo de los negocios enfrenta ambientes que están en constante evolución, lo que provoca cambios permanentes en la distribución de los datos. Si los datos a los cuales le queremos aplicar el modelo son diferentes a los originales que utilizamos para construirlo, la predicción inevitablemente será peor de lo esperado. En riesgo de crédito, por ejemplo, las políticas de otorgamiento (decisiones de umbral) varían en base a razones internas de riesgo y/o factores macroeconómicos, generando un sesgo en la naturaleza de los clientes aceptados. Es importante internalizar los riesgos y limitaciones que conlleva trabajar ambientes cambiantes y hacer un seguimiento de cerca a los resultados del modelo y los nuevos datos.

Si nos permites, nos gustaría dar una última lección de teoría. El monitoreo de modelos se conoce con diferentes nombres, tales como *backtesting* en negocios (principalmente riesgo de crédito) o *dataset shift* en el mundo académico asociado al aprendizaje de máquinas (este término se refiere a los cambios o «quiebres» en la distribución de los datos que pueden ocurrir a lo largo del tiempo).

Para explicar algunos conceptos, asumamos que construimos un modelo de *credit scoring* con datos de 2015 a 2020, y queremos aplicarlo durante 2021 (olvidemos, por ahora, todos los efectos que

puede tener la pandemia del Covid-19). Existen tres tipos de cambios en la distribución de los datos que se deben monitorear con el fin de «levantar alertas» a tiempo y recalibrar los modelos, antes que la pérdida de desempeño traiga consecuencias graves:

- **Cambios en la distribución de las variables explicativas** (las que conforman la matriz X): por ejemplo, que la edad promedio de los clientes pasa de 35 años en el periodo 2015-2020 a 40 años en el año 2021. Una forma para evaluar estos cambios es utilizar una técnica estadística para comparar dos distribuciones, donde la más común es la medida KS (Kolmogorov-Smirnov). Alternativamente, la medida *Population Stability Index* (PSI) fue diseñada precisamente para este propósito, donde se tabulan los cambios en leves, moderados y críticos. Esta medida se construye en base a la divergencia de Kullback-Leibler (KL), que está directamente relacionada con el concepto de entropía, formalizado en el capítulo 6.

  Recomendamos aprovechar los gráficos de caja para mostrar la distribución de los datos en diferentes periodos, y así evaluar gráficamente cambios en las variables. En Bravo & Maldonado (2015) propusimos relacionar los cambios en las variables con los intervalos de confianza que genera una regresión logística. Este intervalo de confianza define un cambio máximo permitido para una variable, por lo que aquellas que se escapan de este rango pueden ser consideradas como variaciones críticas.

  Dos ejemplos de estas visualizaciones pueden verse en la figura 11.1 y figura 11.2, donde cada gráfico representa una variable diferente del modelo que fue construido con datos hasta el año 2004, monitoreado entre los años 2005 y 2007. El desempeño del modelo decrece significativamente (de un AUC=0.63 en 2005 a un AUC=0.55 en 2007), lo que sugiere cambios importantes en la distribución de los datos. La primera figura muestra un cambio leve en la variable de estudio ya que la medida propuesta (DCI) toma un valor 1 cuando no hay cambios con relación a su distribución en el periodo de entrenamiento. En la segunda figura, sin embargo, se observan cambios drásticos ya en el primer

periodo (trimestre 2, 2005). Nuestro método lo identifica con la construcción de intervalos de confianza a 1, 2 y 3 desviaciones estándar 1 ($\overline{Z_\alpha}$= {1,2,3}). La primera evaluación está fuera del intervalo de dos desviaciones, indicando un cambio drástico. Esto explica el deterioro en el desempeño predictivo. Si varias variables presentan este tipo de problemas, lo recomendable es recalibrar y asumir que el modelo quedó obsoleto. Si no hacemos esto a tiempo, el deterioro del desempeño puede ser grave, como se observa en el año 2006, trimestre 2.

Figura 11.1: Monitoreo trimestral de una variable con cambios muy leves

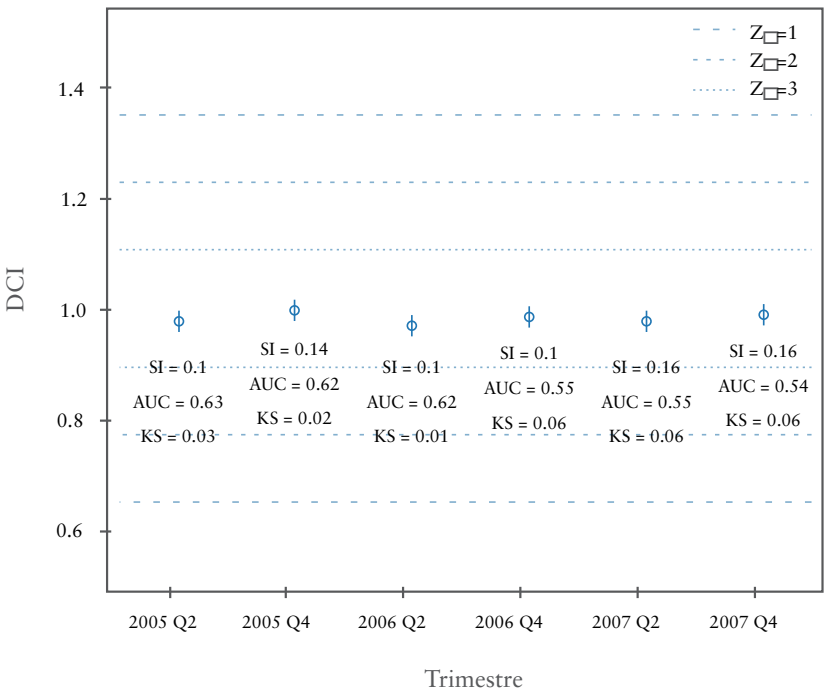
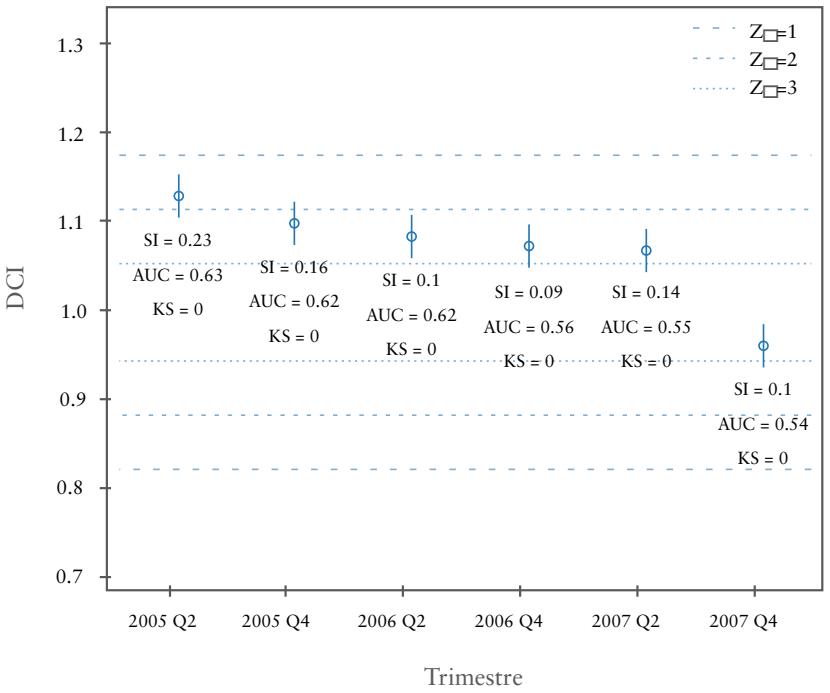


Figura 11.2: Monitoreo trimestral de una variable con cambios drásticos




- **Cambios en la distribución de la variable objetivo** (vector Y): por ejemplo, que la tasa de clientes que caen en *default* suba de un 10% en el periodo 2015-2020 a un 15% en el año 2021. Es recomendable monitorear periódicamente la variable objetivo con el fin de manejar los riesgos y aplicar soluciones oportunamente.
- **Cambios en la relación entre variables explicativas y variable objetivo:** las variables pueden perder su poder predictivo, aunque no ocurran grandes cambios en su distribución. Una variable como la edad puede ser relevante en el periodo 2015-2020 y en el año 2021 ser independiente de la variable que define incumplimiento. Una forma de monitorear estos cambios es utilizar un gráfico similar al de la figura 11.1, pero con una medida de relevancia (como el *test* Chi-cuadrado, ver capítulo 2). Adicionalmente, es muy recomendable monitorear la capacidad predictiva

del modelo mes a mes, utilizando, por ejemplo, la misma medida KS con el fin de contrastar la distribución de la probabilidad de incumplimiento predicha para los clientes «buenos» y «malos».

## 11.2 Mitos y leyendas de la inteligencia artificial

*Droids are not good or bad. They are neutral reflections of those who imprint it*[19].
*Kuiil, Star Wars: The Mandalorian*

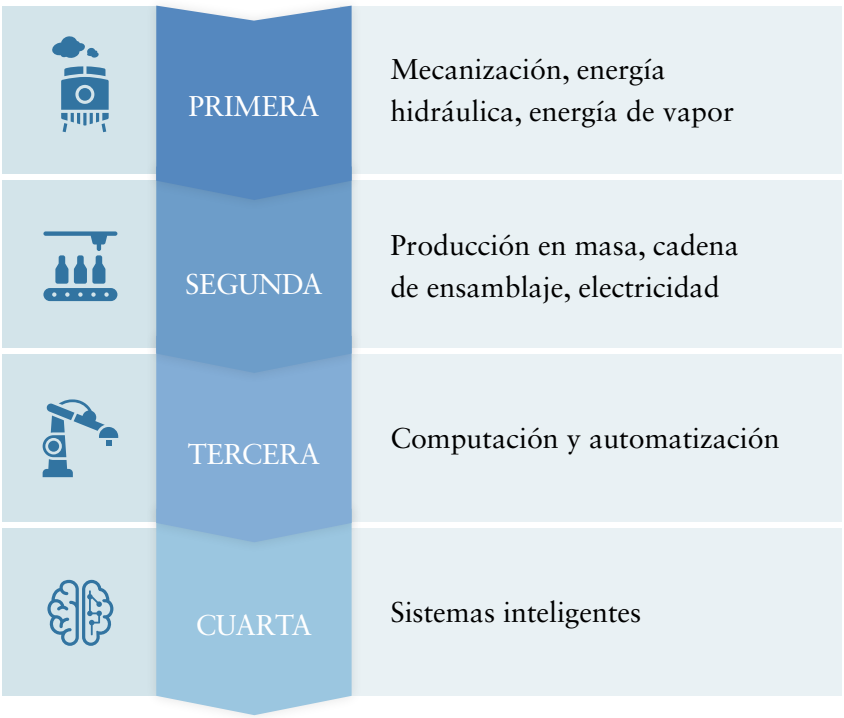
La ciencia ficción y el periodismo han empujado los alcances de la IA hasta dimensiones totalmente inverosímiles. En esta sección discutiremos nuestra visión a futuro de la IA, haciéndonos cargo de algunos mitos detrás de este tema tan popular.

### *11.2.1 La IA nos quitará el trabajo*

Se dice que estamos en la cuarta revolución industrial, que busca hacer las cosas inteligentes *(cognification)* para aumentar la productividad. Si revisamos los datos de revoluciones industriales previas (máquina a vapor, electricidad y producción en masa, computación y automatización; ver figura 11.3), veremos que no ha habido desempleo masivo detrás de ellas. En otras palabras, si alimento con datos una IA para predecir si nos quitarán el trabajo, esta nos dirá que no (y, bueno, hemos visto demasiada ciencia ficción como para dudar de todos modos).

[19] Los *droids* (las IA de la serie) no son ni buenos ni malos; son reflejos neutros de quienes los crean.

Figura 11.3: Las cuatro revoluciones industriales




Tiene sentido asumir el principio básico de la economía detrás de las revoluciones industriales: estas suelen traer consigo mayores ingresos y más tiempo libre. Un mayor ingreso se traduce en un mayor consumo, lo que termina generando nuevas oportunidades de empleo, sobre todo en la industria de servicios y entretención. Es altamente probable que muchos empleos actuales sean automatizados, sin embargo, nacerán muchas nuevas oportunidades laborales que deberían compensar por la pérdida de trabajo. Este fenómeno ya lo estamos viendo en los países industrializados, sin observar un crecimiento masivo del desempleo.

El principal problema es el tiempo de ajuste en la redefinición de los empleos y las jornadas laborales. Las revoluciones industriales anteriores se desarrollaron en un periodo de tiempo mucho más largo que esta. Lo importante es estar preparado como sociedad,

incorporando la IA, y la tecnología en general, en nuestras vidas y las de los más jóvenes.

Debemos ser un equipo con la IA. Si bien se dice que el aprendizaje profundo ha llevado a las máquinas de interpretación de radiografías a ser incluso mejores que los mismos radiólogos, es desfachatado pensar que ellas los llegarán a reemplazar. Al final del día, el mejor radiólogo es aquel que es un equipo con la tecnología, que la utiliza como herramienta para la toma de decisiones.

Un aspecto clave es la educación y la formación de jóvenes. La única forma de estar bien preparados como sociedad para los cambios —inevitables— que se avecinan es estar educados en las nuevas tecnologías. También debe haber un compromiso a nivel país. España, por ejemplo, ha desarrollado una estrategia muy seria de desarrollo de la IA, considerando una fuerte inversión estatal y privada[20]. Los países latinoamericanos vamos mucho más atrás. Esto nos puede traer graves consecuencias, ya que los cambios nos encontrarán «mal parados».

La educación en las nuevas tecnologías parte en casa. Somos de la generación que recuerda cuando llegó la biblioteca digital *Encarta* a las casas, que tuvo consecuencias importantes en la forma de buscar la información. Hoy nuestros niños y jóvenes cuentan con un amplio espectro de herramientas tecnológicas que pueden facilitar la educación interactiva, a los que rara vez se les saca provecho en las escuelas. La tecnología parece más bien una forma de mantener quietos a los niños, más que (mentalmente) inquietos. Debemos alimentar su curiosidad de aprender a través de estas herramientas como parte de su formación.

### *11.2.2 Los avances de la IA nos llevarán a una distopía futurista*

Se dice mucho que estamos cerca de la «singularidad», es decir, que la mente de las IA sea más avanzada que las de los humanos. Se

[20] Estrategia Nacional de Inteligencia Artificial (ENIA), primera versión, noviembre del 2020, disponible en https://portal.mineco.gob.es/RecursosArticulo/mineco/ministerio/ficheros/201202_ENIA_V1_0.pdf

dice también que en ese momento las IA se «darán cuenta» que la humanidad puede ser esclavizada por ellas. Sentimos que estamos muy lejos de que pase algo así.

Es importante comprender la diferencia de dos conceptos que han llevado a muchas confusiones. Por un lado, IA se refiere a dar a máquinas capacidades humanas, haciéndolas «inteligentes». Es un concepto más bien genérico. Por otro lado, aprendizaje de máquinas (ML) se refiere a algoritmos que «aprenden» de los datos sin estar explícitamente programados para ello. Pensemos, por ejemplo, en el juego del «gato» (*tic-tac-toe* o tres en línea). Es posible programar todas las reglas que le permitirían a una máquina dominar el juego, pero la gracia de ML es que no necesitamos hacer eso, sino utilizar un enfoque de prueba y error. La idea es hacer jugar a la máquina el juego sin decirle qué hacer, más que debe colocar un círculo o una cruz cuando le corresponda. Al principio jugará al azar y perderá, pero si le decimos cuándo pierde y cuándo gana, la máquina «aprenderá» y será capaz de generar las reglas que le permitan dominar el juego (sin haberla programado para eso).

Los algoritmos de ML utilizan técnicas como las ANN vistas en el capítulo 8, pero decir que estas técnicas emulan un cerebro humano es un exceso. La analogía aceptada es que las redes artificiales son similares a un cerebro muy primitivo, como el de un gusano de tierra. El tema es que las IA utilizan ML para dar capacidades humanas a máquinas (con cerebros de gusano de tierra), por lo que ambos conceptos se terminaron confundiendo y fusionando. Una IA compleja, como un automóvil que se maneja solo, no es un cerebro más avanzado, sino que una serie de «cerebritos» de gusano trabajando en conjunto, reconociendo objetos y tomando decisiones de tránsito.

Los grandes avances en ML han abordado aspectos muy específicos que nos permiten acercarnos con pasos de bebé a cerebros un poco más sofisticados que los de un gusano de tierra. Sin embargo, aún estamos muy lejos de modelar aspectos como conciencia humana o libre albedrío. Esto no se debe necesariamente a nuestra incapacidad como «modeladores de la realidad»: la misma neurociencia no ha sido capaz de comprender este tipo de comportamientos

humanos, ¿cómo podríamos modelar matemáticamente algo que no entendemos? Además, existen pocos científicos de IA tratando de abordar desafíos como darle conciencia a las IA (por más que la ciencia ficción nos haga creer lo contrario).

Otro tema que parece ralentizar esta visión futurista de las películas son los avances de la robótica, que nos parecen más lentos de lo esperado. Considerando que había robots que jugaban fútbol hace ya más de 20 años, uno hubiese esperado, a estas alturas, equipos de robots ágiles y capaces de vencer a humanos. Si bien existen robots bastante impresionantes, parecen pasar muchos años como prueba de concepto y funcionan de forma óptima solo bajo condiciones de laboratorio.

### *11.2.3 Entonces, ¿está todo bien?*

Bueno, no, solo desmitificamos algunos aspectos de la IA motivados por la ciencia ficción. Es cierto que la IA y la robótica pueden ser utilizadas como armamento *(weaponized)*, obedeciendo las órdenes de quienes lo construyen y no rebelándose contra ellos. Que existan guerras peleadas por IA parece mucho más plausible a que hagan cosas para las cuales no fueron diseñadas.

Otro mito que consideramos totalmente válido es que la IA va a aumentar las brechas de desigualdad, tanto entre países como dentro del país. Esta es una de las principales preocupaciones del área de investigación asociada a la ética en IA.

El hecho de que haya países líderes en el desarrollo de IA causa una desigual formación en el área, también una fuga de talentos. De forma similar, hablamos de una adopción tecnológica dentro de las familias, pero esto es posible solo en aquellas familias donde la tecnología de punta está disponible. La tecnología tiene un costo y su acceso es desigual, al igual que la capacidad de los padres de formar a los hijos en un ambiente altamente tecnologizado.

Los mismos modelos matemáticos que utilizamos en la actualidad terminan perpetuando un círculo vicioso de desigualdad. Esto no se debe a que sean «malvados por naturaleza», sino porque se

construyen con el único objetivo de maximizar utilidades. Entidades reguladoras muchas veces prohíben el uso de variables que llevan a discriminación, como género o etnia, sin embargo, una IA puede encontrar *proxies* para estas variables (en Estados Unidos, por ejemplo, el código *ZIP* puede separar barrios de afroamericanos, hispanos y caucásicos sin la necesidad de usar una variable como etnia).

Insistimos en el hecho que las IA cumplen el propósito de sus creadores. Es fácil esconderse en argumentos tipo «caja negra» como excusa para responsabilizar a una IA de las decisiones que toman las empresas y otras instituciones. Por suerte, este argumento pierde cada vez más validez con los avances en ML interpretable.

Un punto a favor de la automatización de decisiones desarrollado en el capítulo 10 es que podemos imponer «restricciones sociales» a los modelos. Por ejemplo, a un modelo de crédito que maximice las utilidades en base a medidas de *profit* se le pueden incorporar restricciones que aseguren ciertas cuotas a segmentos puntuales con el fin de llegar a un acceso más equitativo de crédito. Esto es posible de desarrollar con el nivel de la ciencia en IA actual, y es mucho más efectivo que simplemente ignorar ciertas variables sabiendo que se pueden usar otras, altamente correlacionadas con las prohibidas. Lo que falta es que como sociedad nos pongamos de acuerdo en qué queremos exigirles a las empresas que tienen un impacto social mediante la utilización de estas tecnologías, y que se creen las entidades que velen por que estas exigencias se cumplan.

## 11.3 Recursos adicionales

¿Te gustaron los tópicos vistos en este libro? ¿Alguno en particular? Te invitamos a profundizar dependiendo de tus intereses. A continuación, destacamos una serie de recursos, algunos incluso gratuitos, que nos ayudaron a conformar este libro y que, creemos, pueden serte de mucha utilidad (principalmente en inglés, ya que lamentablemente no hay una gran variedad de libros en español al respecto):

- Para profundizar más en las técnicas predictivas y descriptivas, recomendamos *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* de T. Hastie, R. Tibshirani & J. Friedman, el cual está disponible de forma gratuita. Este libro es matemáticamente desafiante, pero permite llegar a un nivel avanzado en la caja de herramientas que hemos construido.
- Un tema no cubierto en el libro anterior es aprendizaje profundo. Para sumergirse en este tema, podemos recomendar los libros *Deep Learning,* por I. Goodfellow, Y. Bengio & A. Courville, y también *Neural Networks and Deep Learning,* por C. C. Aggarwal.
- Otro aspecto que no está cubierto en el primer libro (o muy tangencialmente) es el pronóstico vía series de tiempo. Podemos recomendar el libro (gratuito) de R. Hyndman & G. Athanasopoulos titulado *Forecasting: Principles and Practice*.
- Otro libro gratuito que aborda aspectos metodológicos es *Interpretable Machine Learning* por C. Molnar. Este manuscrito se enfoca en aspectos de interpretación de modelos predictivos, algo muy relevante en el área de negocios.
- Los libros anteriores no profundizan en aspectos de negocios. Sin embargo, existen varios enfocados en las aplicaciones discutidas aquí:

    - Riesgo de crédito es un área muy estudiada con varios libros disponibles, siendo, probablemente, el más reciente: *Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS* por B. Baesens, D. Roesch & H. Scheule.
    - Existen también varios libros en *marketing analytics* que profundizan en temas como perfilamiento o fuga de clientes. Nuestro favorito es *Database Marketing: Analyzing and Managing Customers,* de R. C. Blattberg, B. -D. Kim & S. A. Neslin.
    - El libro *Profit Driven Business Analytics: A Practitioner's Guide to Transforming* Big Data *into Added Value* de W. Verbeke, B. Baesens & C. Bravo, es una excelente guía para profundizar en aspectos como medidas de *profit* e inferencia

causal (medidas de *uplift*) destacadas en el capítulo 10, además de reforzar aspectos de modelación predictiva.
- Un libro que permite profundizar en aspectos como bases de datos y Big Data más cercanos a la empresa (la mayoría de los recursos suelen ser muy orientados al *software* y no a la aplicación) es *Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data* de W. Lemahieu, S. vanden Broucke & B. Baesens.
- Finalmente, el libro *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection,* de B. Baesens, V. Van Vlasselaer & W. Verbeke, explora las técnicas de analítica con foco en detección de fraude.

- Existe, además, una serie de libros de «lectura *light*», más enfocados en motivar la importancia de la analítica de negocios y la ciencia de datos que de abordar las metodologías con profundidad. Algunos libros recomendados, por sus historias y calidad narrativa, son *The Signal and the Noise: Why So Many Predictions Fail-but Some Don't* de N. Silver (probablemente el único *best seller* en el área, del que existe una versión en español) y *Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die* de E. Siegel.

## Referencias

Baesens, B., Van Vlasselaer, V. & Verbeke, W. (2015). *Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection*. New Jersey: John Wiley and Sons.

Bravo, C. & Maldonado, S. (2015). «Fieller's Stability Measure: A Novel Model-dependent Backtesting Approach». *Journal of the Operational Research Society,* 66, 1895-1905.

Venkatesan, R., Farris, P. & Wilcox, R. (2015). *Cutting Edge Marketing Analytics: Real World Cases and Data Sets for Hands On Learning*. New Jersey: Pearson.

Este libro se terminó de imprimir
en Santiago de Chile,
diciembre de 2021

Teléfono: 22 22 38 100 / ril@rileditores.com

Se utilizó tecnología de última generación que reduce
el impacto medioambiental, pues ocupa estrictamente
el papel necesario para su producción, y se aplicaron
altos estándares para la gestión y reciclaje de desechos
en toda la cadena de producción.

Este libro constituye la primera guía en español que estudia las competencias clave para crear valor en base a datos de empresas. No solo presenta una perspectiva general de las técnicas y aplicaciones asociadas a la analítica de negocios, sino que también incluye laboratorios paso a paso para la implementación de estas herramientas. Los códigos se desarrollan en Python, que es la principal plataforma gratuita para la construcción de modelos analíticos.

Los autores ofrecen cobertura esencial a los principales temas actuales:

- Big Data (Hadoop, Spark, MapReduce y otros)
- Aprendizaje profundo
- Perfilamiento de clientes y marketing basado en datos
- Financial Analytics y modelación predictiva de riesgo
- Analytics con datos avanzados (textos e imágenes)

El libro está dirigido a analistas o consultores que deseen profundizar en técnicas avanzadas de analítica de negocios. A su vez, puede resultar muy interesante para estadísticos e ingenieros que deseen aplicar conocimientos en el mundo de los negocios, pues combina de forma óptima la teoría con ejemplos aplicados y la experiencia de los autores.

Si bien puede parecer avanzado, no requiere conocimientos previos más que experiencia básica de análisis de datos y herramientas computacionales. Además, su diseño amigable facilita enormemente el aprendizaje.

Los datos y códigos del libro se encuentran en:
*www.vairetti-maldonado.cl*

ISBN 978-84-18982-63-7

RiL editores